The Bugzilla Guide - 4.4.12+ Release

Contents

Abstract

This is the documentation for Bugzilla, a bug-tracking system from mozilla.org. Bugzilla is an enterprise-class piece of software that tracks millions of bugs and issues for hundreds of organizations around the world.

The most current version of this document can always be found on the Bugzilla Documentation Page[1].

[1] http://www.bugzilla.org/docs/

Chapter 1

About This Guide

Copyright Information

Disclaimer

No liability for the contents of this document can be accepted. Follow the instructions herein at your own risk. This document may contain errors and inaccuracies that may damage your system, cause your partner to leave you, your boss to fire you, your cats to pee on your furniture and clothing, and global thermonuclear war. Proceed with caution.

Naming of particular products or brands should not be seen as endorsements, with the exception of the term "GNU/Linux". We wholeheartedly endorse the use of GNU/Linux; it is an extremely versatile, stable, and robust operating system that offers an ideal operating environment for Bugzilla.

Although the Bugzilla development team has taken great care to ensure that all exploitable bugs have been fixed, security holes surely exist in any piece of code. Great care should be taken both in the installation and usage of this software. The Bugzilla development team members assume no liability for your use of Bugzilla. You have the source code, and are responsible for auditing it yourself to ensure your security needs are met.

New Versions

This is the 4.4.12+ version of The Bugzilla Guide. It is so named to match the current version of Bugzilla.

The latest version of this guide can always be found at http://www.bugzilla.org/docs/. However, you should read the version which came with the Bugzilla release you are using.

In addition, there are Bugzilla template localization projects in several languages[1]. They may have translated documentation available. If you would like to volunteer to translate the Guide into additional languages, please visit the Bugzilla L10n team[2] page.

[1] http://www.bugzilla.org/download/#localizations
[2] https://wiki.mozilla.org/Bugzilla:L10n

Credits

The people listed below have made enormous contributions to the creation of this Guide, through their writing, dedicated hacking efforts, numerous e-mail and IRC support sessions, and overall excellent contribution to the Bugzilla community:

Matthew P. Barnson mbarnson@sisna.com for the Herculean task of pulling together the Bugzilla Guide and shepherding it to 2.14.

Terry Weissman terry@mozilla.org for initially writing Bugzilla and creating the README upon which the UNIX installation documentation is largely based.

Tara Hernandez tara@tequilarists.org for keeping Bugzilla development going strong after Terry left mozilla.org and for running landfill.

Dave Lawrence dkl@redhat.com for providing insight into the key differences between Red Hat's customized Bugzilla.

Dawn Endico endico@mozilla.org for being a hacker extraordinaire and putting up with Matthew's incessant questions and arguments on irc.mozilla.org in #mozwebtools

Jacob Steenhagen jake@bugzilla.org for taking over documentation during the 2.17 development period.

Dave Miller justdave@bugzilla.org for taking over as project lead when Tara stepped down and continually pushing for the documentation to be the best it can be.

Thanks also go to the following people for significant contributions to this documentation: Kevin Brannen, Vlad Dascalu, Ben FrantzDale, Eric Hanson, Zach Lipton, Gervase Markham, Andrew Pearson, Joe Robins, Spencer Smith, Ron Teitelbaum, Shane Travis, Martin Wulffeld.

Also, thanks are due to the members of the mozilla.support.bugzilla[3] newsgroup (and its predecessor, netscape.public.mozilla.webtools). Without your discussions, insight, suggestions, and patches, this could never have happened.

Document Conventions

This document uses the following conventions:

Caution
This is a caution. Make sure to read this to not be in trouble!

Tip
This is a hint or tip, especially about some configuration tweaks.

Note
This is just a note, for your information.

Warning
This is a warning, something you should take care of.

[3] news://news.mozilla.org/mozilla.support.bugzilla

A filename or a path to a filename is displayed like this: `/path/to/filename.ext`

A command to type in the shell is displayed like this: **command --arguments**

bash$ represents a normal user's prompt under bash shell

bash# represents a root user's prompt under bash shell

A word which is in the glossary will appear like this: *Bugzilla*

A sample of code is illustrated like this:

```
First Line of Code
Second Line of Code
...
```

This documentation is maintained in DocBook 4.2 XML format. Changes are best submitted as plain text or XML diffs, attached to a bug filed in the Bugzilla Documentation[4] component.

[4] http://bugzilla.mozilla.org/enter_bug.cgi?product=Bugzilla;component=Documentation

Chapter 2

Installing Bugzilla

Installation

Note
If you just want to *use* Bugzilla, you do not need to install it. None of this chapter is relevant to you. Ask your Bugzilla administrator for the URL to access it from your web browser.

The Bugzilla server software is usually installed on Linux or Solaris. If you are installing on another OS, check Section 2.5 before you start your installation to see if there are any special instructions.

This guide assumes that you have administrative access to the Bugzilla machine. It not possible to install and run Bugzilla itself without administrative access except in the very unlikely event that every single prerequisite is already installed.

Warning
The installation process may make your machine insecure for short periods of time. Make sure there is a firewall between you and the Internet.

You are strongly recommended to make a backup of your system before installing Bugzilla (and at regular intervals thereafter :-).

In outline, the installation proceeds as follows:

1. Install Perl (5.8.1 or above)
2. Install a Database Engine
3. Install a Webserver
4. Install Bugzilla
5. Install Perl modules
6. Install a Mail Transfer Agent (Sendmail 8.7 or above, or an MTA that is Sendmail-compatible with at least this version)
7. Configure all of the above.

Perl

Installed Version Test:

```
perl -v
```

Any machine that doesn't have Perl on it is a sad machine indeed. If you don't have it and your OS doesn't provide official packages, visit http://www.perl.org. Although Bugzilla runs with Perl 5.8.1, it's a good idea to be using the latest stable version.

Database Engine

Bugzilla supports MySQL, PostgreSQL and Oracle as database servers. You only require one of these systems to make use of Bugzilla.

MySQL

Installed Version Test:

```
mysql -V
```

If you don't have it and your OS doesn't provide official packages, visit http://www.mysql.com. You need MySQL version 5.0.15 or higher.

Note
Many of the binary versions of MySQL store their data files in `/var`. On some Unix systems, this is part of a smaller root partition, and may not have room for your bug database. To change the data directory, you have to build MySQL from source yourself, and set it as an option to `configure`.

If you install from something other than a packaging/installation system, such as .rpm (RPM Package Manager), .deb (Debian Package), .exe (Windows Executable), or .msi (Windows Installer), make sure the MySQL server is started when the machine boots.

PostgreSQL

Installed Version Test:

```
psql -V
```

If you don't have it and your OS doesn't provide official packages, visit http://www.postgresql.org/. You need PostgreSQL version 8.03.0000 or higher.

If you install from something other than a packaging/installation system, such as .rpm (RPM Package Manager), .deb (Debian Package), .exe (Windows Executable), or .msi (Windows Installer), make sure the PostgreSQL server is started when the machine boots.

Oracle

Installed Version Test:

```
select * from v$version
```

(you first have to log in into your DB)

If you don't have it and your OS doesn't provide official packages, visit http://www.oracle.com/. You need Oracle version 10.02.0 or higher.

If you install from something other than a packaging/installation system, such as .rpm (RPM Package Manager), .deb (Debian Package), .exe (Windows Executable), or .msi (Windows Installer), make sure the Oracle server is started when the machine boots.

Web Server

Installed Version Test: view the default welcome page at http://<your-machine>/

You have freedom of choice here, pretty much any web server that is capable of running *CGI* scripts will work. However, we strongly recommend using the Apache web server (either 1.3.x or 2.x), and the installation instructions usually assume you are using it. If you have got Bugzilla working using another web server, please share your experiences with us by filing a bug in Bugzilla Documentation[1].

If you don't have Apache and your OS doesn't provide official packages, visit `http://httpd.apache.org/`.

Bugzilla

Download a Bugzilla tarball[2] (or check it out from Bzr[3]) and place it in a suitable directory, accessible by the default web server user (probably "apache" or "www"). Good locations are either directly in the web server's document directories or in `/usr/local` with a symbolic link to the web server's document directories or an alias in the web server's configuration.

Caution
The default Bugzilla distribution is NOT designed to be placed in a `cgi-bin` directory. This includes any directory which is configured using the `ScriptAlias` directive of Apache.

Once all the files are in a web accessible directory, make that directory writable by your web server's user. This is a temporary step until you run the `checksetup.pl` script, which locks down your installation.

Perl Modules

Bugzilla's installation process is based on a script called `checksetup.pl`. The first thing it checks is whether you have appropriate versions of all the required Perl modules. The aim of this section is to pass this check. When it passes, proceed to Section 2.2.

At this point, you need to `su` to root. You should remain as root until the end of the install. To check you have the required modules, run:

```
bash# ./checksetup.pl --check-modules
```

`checksetup.pl` will print out a list of the required and optional Perl modules, together with the versions (if any) installed on your machine. The list of required modules is reasonably long; however, you may already have several of them installed.

The preferred way to install missing Perl modules is to use the package manager provided by your operating system (e.g "rpm" or "yum" on Linux distros, or "ppm" on Windows if using ActivePerl, see Section 2.5.1.2). If some Perl modules are still missing or are too old, then we recommend using the `install-module.pl` script (doesn't work with ActivePerl on Windows). If for some reason you really need to install the Perl modules manually, see Appendix C. For instance, on Unix, you invoke `install-module.pl` as follows:

```
bash# perl install-module.pl <modulename>
```

Tip
Many people complain that Perl modules will not install for them. Most times, the error messages complain that they are missing a file in "@INC". Virtually every time, this error is due to permissions being set too restrictively for you to compile Perl modules or not having the necessary Perl development libraries installed on your system. Consult your local UNIX systems administrator for help solving these permissions issues; if you *are* the local UNIX sysadmin, please consult the newsgroup/mailing list for further assistance or hire someone to help you out.

[1] `http://bugzilla.mozilla.org/enter_bug.cgi?product=Bugzilla;component=Documentation`
[2] `http://www.bugzilla.org/download/`
[3] `https://wiki.mozilla.org/Bugzilla:Bzr`

Note
If you are using a package-based system, and attempting to install the Perl modules from CPAN, you may need to install the "development" packages for MySQL and GD before attempting to install the related Perl modules. The names of these packages will vary depending on the specific distribution you are using, but are often called `<packagename>-devel`.

Here is a complete list of modules and their minimum versions. Some modules have special installation notes, which follow.

Required Perl modules:

1. CGI (3.51)
2. Date::Format (2.23)
3. DateTime (0.28)
4. DateTime::TimeZone (0.71)
5. DBI (1.614)
6. DBD::mysql (4.001) if using MySQL
7. DBD::Pg (2.7.0) if using PostgreSQL
8. DBD::Oracle (1.19) if using Oracle
9. Digest::SHA (any)
10. Email::Send (2.04)
11. Email::MIME (1.904)
12. Template (2.22)
13. URI (1.37)

Optional Perl modules:

1. GD (1.20) for bug charting
2. Template::Plugin::GD::Image (any) for Graphical Reports
3. Chart::Lines (2.1.0) for bug charting
4. GD::Graph (any) for bug charting
5. GD::Text (any) for bug charting
6. XML::Twig (any) for bug import/export
7. MIME::Parser (5.406) for bug import/export
8. LWP::UserAgent (any) for Automatic Update Notifications
9. PatchReader (0.9.6) for pretty HTML view of patches
10. Net::LDAP (any) for LDAP Authentication
11. Authen::SASL (any) for SASL Authentication
12. Authen::Radius (any) for RADIUS Authentication
13. SOAP::Lite (0.712) for the web service interface
14. JSON::RPC (any) for the JSON-RPC interface

15. Test::Taint (any) for the web service interface
16. HTML::Parser (3.67) for More HTML in Product/Group Descriptions
17. HTML::Scrubber (any) for More HTML in Product/Group Descriptions
18. Email::Reply (any) for Inbound Email
19. TheSchwartz (1.07) for Mail Queueing
20. Daemon::Generic (any) for Mail Queueing
21. mod_perl2 (1.999022) for mod_perl

Mail Transfer Agent (MTA)

Bugzilla is dependent on the availability of an e-mail system for its user authentication and for other tasks.

Note
This is not entirely true. It is possible to completely disable email sending, or to have Bugzilla store email messages in a file instead of sending them. However, this is mainly intended for testing, as disabling or diverting email on a production machine would mean that users could miss important events (such as bug changes or the creation of new accounts).
For more information, see the "mail_delivery_method" parameter in Section 3.1.

On Linux, any Sendmail-compatible MTA (Mail Transfer Agent) will suffice. Sendmail, Postfix, qmail and Exim are examples of common MTAs. Sendmail is the original Unix MTA, but the others are easier to configure, and therefore many people replace Sendmail with Postfix or Exim. They are drop-in replacements, so Bugzilla will not distinguish between them.

If you are using Sendmail, version 8.7 or higher is required. If you are using a Sendmail-compatible MTA, it must be congruent with at least version 8.7 of Sendmail.

Consult the manual for the specific MTA you choose for detailed installation instructions. Each of these programs will have their own configuration files where you must configure certain parameters to ensure that the mail is delivered properly. They are implemented as services, and you should ensure that the MTA is in the auto-start list of services for the machine.

If a simple mail sent with the command-line 'mail' program succeeds, then Bugzilla should also be fine.

Installing Bugzilla on mod_perl

It is now possible to run the Bugzilla software under `mod_perl` on Apache. `mod_perl` has some additional requirements to that of running Bugzilla under `mod_cgi` (the standard and previous way).

Bugzilla requires `mod_perl` to be installed, which can be obtained from `http://perl.apache.org` - Bugzilla requires version 1.999022 (AKA 2.0.0-RC5) to be installed.

Configuration

Warning
Poorly-configured MySQL and Bugzilla installations have given attackers full access to systems in the past. Please take the security parts of these guidelines seriously, even for Bugzilla machines hidden away behind your firewall. Be certain to read Chapter 4 for some important security tips.

localconfig

You should now run `checksetup.pl` again, this time without the `--check-modules` switch.

```
bash# ./checksetup.pl
```

This time, `checksetup.pl` should tell you that all the correct modules are installed and will display a message about, and write out a file called, `localconfig`. This file contains the default settings for a number of Bugzilla parameters.

Load this file in your editor. The only two values you *need* to change are $db_driver and $db_pass, respectively the type of the database and the password for the user you will create for your database. Pick a strong password (for simplicity, it should not contain single quote characters) and put it here. $db_driver can be either 'mysql', 'Pg', 'Oracle' or 'Sqlite'.

Note
In Oracle, `$db_name` should actually be the SID name of your database (e.g. "XE" if you are using Oracle XE).

You may need to change the value of *webservergroup* if your web server does not run in the "apache" group. On Debian, for example, Apache runs in the "www-data" group. If you are going to run Bugzilla on a machine where you do not have root access (such as on a shared web hosting account), you will need to leave *webservergroup* empty, ignoring the warnings that `checksetup.pl` will subsequently display every time it is run.

Caution
If you are using suexec, you should use your own primary group for *webservergroup* rather than leaving it empty, and see the additional directions in the suexec section Section 2.6.6.1.

The other options in the `localconfig` file are documented by their accompanying comments. If you have a slightly non-standard database setup, you may wish to change one or more of the other "$db_*" parameters.

Database Server

This section deals with configuring your database server for use with Bugzilla. Currently, MySQL (Section 2.2.2.2), PostgreSQL (Section 2.2.2.3), Oracle (Section 2.2.2.4) and SQLite (Section 2.2.2.5) are available.

Bugzilla Database Schema

The Bugzilla database schema is available at Ravenbrook[4]. This very valuable tool can generate a written description of the Bugzilla database schema for any version of Bugzilla. It can also generate a diff between two versions to help someone see what has changed.

MySQL

Caution
MySQL's default configuration is insecure. We highly recommend to run `mysql_secure_installation` on Linux or the MySQL installer on Windows, and follow the instructions. Important points to note are:

1. Be sure that the root account has a secure password set.
2. Do not create an anonymous account, and if it exists, say "yes" to remove it.
3. If your web server and MySQL server are on the same machine, you should disable the network access.

[4] http://www.ravenbrook.com/project/p4dti/tool/cgi/bugzilla-schema/

Allow large attachments and many comments

By default, MySQL will only allow you to insert things into the database that are smaller than 1MB. Attachments may be larger than this. Also, Bugzilla combines all comments on a single bug into one field for full-text searching, and the combination of all comments on a single bug could in some cases be larger than 1MB.

To change MySQL's default, you need to edit your MySQL configuration file, which is usually `/etc/my.cnf` on Linux. We recommend that you allow at least 4MB packets by adding the "max_allowed_packet" parameter to your MySQL configuration in the "[mysqld]" section, like this:

```
[mysqld]
# Allow packets up to 4MB
max_allowed_packet=4M
```

Allow small words in full-text indexes

By default, words must be at least four characters in length in order to be indexed by MySQL's full-text indexes. This causes a lot of Bugzilla specific words to be missed, including "cc", "ftp" and "uri".

MySQL can be configured to index those words by setting the ft_min_word_len param to the minimum size of the words to index. This can be done by modifying the `/etc/my.cnf` according to the example below:

```
[mysqld]
# Allow small words in full-text indexes
ft_min_word_len=2
```

Rebuilding the indexes can be done based on documentation found at http://www.mysql.com/doc/en/Fulltext_Fine-tuning.html.

Add a user to MySQL

You need to add a new MySQL user for Bugzilla to use. (It's not safe to have Bugzilla use the MySQL root account.) The following instructions assume the defaults in `localconfig`; if you changed those, you need to modify the SQL command appropriately. You will need the *$db_pass* password you set in `localconfig` in Section 2.2.1.

We use an SQL **GRANT** command to create a "bugs" user. This also restricts the "bugs"user to operations within a database called "bugs", and only allows the account to connect from "localhost". Modify it to reflect your setup if you will be connecting from another machine or as a different user.

Run the `mysql` command-line client and enter:

```
mysql> GRANT SELECT, INSERT,
       UPDATE, DELETE, INDEX, ALTER, CREATE, LOCK TABLES,
       CREATE TEMPORARY TABLES, DROP, REFERENCES ON bugs.*
       TO bugs@localhost IDENTIFIED BY '$db_pass';
mysql> FLUSH PRIVILEGES;
```

Permit attachments table to grow beyond 4GB

By default, MySQL will limit the size of a table to 4GB. This limit is present even if the underlying filesystem has no such limit. To set a higher limit, follow these instructions.

After you have completed the rest of the installation (or at least the database setup parts), you should run the `MySQL` command-line client and enter the following, replacing `$bugs_db` with your Bugzilla database name (*bugs* by default):

```
mysql> use $bugs_db
mysql> ALTER TABLE attachments
           AVG_ROW_LENGTH=1000000, MAX_ROWS=20000;
```

The above command will change the limit to 20GB. Mysql will have to make a temporary copy of your entire table to do this. Ideally, you should do this when your attachments table is still small.

Note
This does not affect Big Files, attachments that are stored directly on disk instead of in the database.

PostgreSQL

Add a User to PostgreSQL

You need to add a new user to PostgreSQL for the Bugzilla application to use when accessing the database. The following instructions assume the defaults in `localconfig`; if you changed those, you need to modify the commands appropriately. You will need the *`$db_pass`* password you set in `localconfig` in Section 2.2.1.

On most systems, to create the user in PostgreSQL, you will need to login as the root user, and then

```
bash# su - postgres
```

As the postgres user, you then need to create a new user:

```
bash$ createuser -U postgres -dRSP bugs
```

When asked for a password, provide the password which will be set as *`$db_pass`* in `localconfig`. The created user will not be a superuser (-S) and will not be able to create new users (-R). He will only have the ability to create databases (-d).

Configure PostgreSQL

Now, you will need to edit `pg_hba.conf` which is usually located in `/var/lib/pgsql/data/`. In this file, you will need to add a new line to it as follows:

```
host all bugs 127.0.0.1 255.255.255.255 md5
```

This means that for TCP/IP (host) connections, allow connections from '127.0.0.1' to 'all' databases on this server from the 'bugs' user, and use password authentication (md5) for that user.

Now, you will need to restart PostgreSQL, but you will need to fully stop and start the server rather than just restarting due to the possibility of a change to `postgresql.conf`. After the server has restarted, you will need to edit `localconfig`, finding the `$db_driver` variable and setting it to `Pg` and changing the password in `$db_pass` to the one you picked previously, while setting up the account.

Oracle

Create a New Tablespace

You can use the existing tablespace or create a new one for Bugzilla. To create a new tablespace, run the following command:

```
CREATE TABLESPACE bugs
DATAFILE '$path_to_datafile' SIZE 500M
AUTOEXTEND ON NEXT 30M MAXSIZE UNLIMITED
```

Here, the name of the tablespace is 'bugs', but you can choose another name. *`$path_to_datafile`* is the path to the file containing your database, for instance `/u01/oradata/bugzilla.dbf`. The initial size of the database file is set in this example to 500 Mb, with an increment of 30 Mb everytime we reach the size limit of the file.

Add a User to Oracle

The user name and password must match what you set in `localconfig` (`$db_user` and `$db_pass`, respectively). Here, we assume that the user name is 'bugs' and the tablespace name is the same as above.

```
CREATE USER bugs
IDENTIFIED BY "$db_pass"
DEFAULT TABLESPACE bugs
TEMPORARY TABLESPACE TEMP
PROFILE DEFAULT;
-- GRANT/REVOKE ROLE PRIVILEGES
GRANT CONNECT TO bugs;
GRANT RESOURCE TO bugs;
-- GRANT/REVOKE SYSTEM PRIVILEGES
GRANT UNLIMITED TABLESPACE TO bugs;
GRANT EXECUTE ON CTXSYS.CTX_DDL TO bugs;
```

Configure the Web Server

If you use Apache, append these lines to `httpd.conf` to set ORACLE_HOME and LD_LIBRARY_PATH. For instance:

```
SetEnv ORACLE_HOME /u01/app/oracle/product/10.2.0/
SetEnv LD_LIBRARY_PATH /u01/app/oracle/product/10.2.0/lib/
```

When this is done, restart your web server.

SQLite

Caution
Due to SQLite's concurrency limitations[a] we recommend SQLite only for small and development Bugzilla installations.

[a] http://sqlite.org/faq.html#q5

No special configuration is required to run Bugzilla on SQLite. The database will be stored in `data/db/$db_name`, where `$db_name` is the database name defined in `localconfig`.

checksetup.pl

Next, rerun `checksetup.pl`. It reconfirms that all the modules are present, and notices the altered localconfig file, which it assumes you have edited to your satisfaction. It compiles the UI templates, connects to the database using the 'bugs' user you created and the password you defined, and creates the 'bugs' database and the tables therein.

After that, it asks for details of an administrator account. Bugzilla can have multiple administrators - you can create more later - but it needs one to start off with. Enter the email address of an administrator, his or her full name, and a suitable Bugzilla password.

`checksetup.pl` will then finish. You may rerun `checksetup.pl` at any time if you wish.

Web server

Configure your web server according to the instructions in the appropriate section. (If it makes a difference in your choice, the Bugzilla Team recommends Apache.) To check whether your web server is correctly configured, try to access `testagent.cgi` from your web server. If "OK" is displayed, then your configuration is successful. Regardless of which web server you are using, however, ensure that sensitive information is not remotely available by properly applying the access controls in Section 4.2.1. You can run `testserver.pl` to check if your web server serves Bugzilla files as expected.

Bugzilla using Apache

You have two options for running Bugzilla under Apache - mod_cgi (the default) and mod_perl (new in Bugzilla 2.23)

Apache httpd with mod_cgi

To configure your Apache web server to work with Bugzilla while using mod_cgi, do the following:

1. Load `httpd.conf` in your editor. In Fedora and Red Hat Linux, this file is found in `/etc/httpd/conf`.
2. Apache uses `<Directory>` directives to permit fine-grained permission setting. Add the following lines to a directive that applies to the location of your Bugzilla installation. (If such a section does not exist, you'll want to add one.) In this example, Bugzilla has been installed at `/var/www/html/bugzilla`.

```
<Directory /var/www/html/bugzilla>
AddHandler cgi-script .cgi
Options +ExecCGI
DirectoryIndex index.cgi index.html
AllowOverride All
</Directory>
```

   These instructions: allow apache to run .cgi files found within the bugzilla directory; instructs the server to look for a file called `index.cgi` or, if not found, `index.html` if someone only types the directory name into the browser; and allows Bugzilla's `.htaccess` files to override some global permissions.

   Note
   It is possible to make these changes globally, or to the directive controlling Bugzilla's parent directory (e.g. `<Directory /var/www/html/>`). Such changes would also apply to the Bugzilla directory... but they would also apply to many other places where they may or may not be appropriate. In most cases, including this one, it is better to be as restrictive as possible when granting extra access.

   Note
   On Windows, you may have to also add the `ScriptInterpreterSource Registry-Strict` line, see Windows specific notes.

3. `checksetup.pl` can set tighter permissions on Bugzilla's files and directories if it knows what group the web server runs as. Find the `Group` line in `httpd.conf`, place the value found there in the *`$webservergroup`* variable in `localconfig`, then rerun `checksetup.pl`.
4. Optional: If Bugzilla does not actually reside in the webspace directory, but instead has been symbolically linked there, you will need to add the following to the `Options` line of the Bugzilla `<Directory>` directive (the same one as in the step above):

```
+FollowSymLinks
```

   Without this directive, Apache will not follow symbolic links to places outside its own directory structure, and you will be unable to run Bugzilla.

Apache httpd with mod_perl

Some configuration is required to make Bugzilla work with Apache and mod_perl

1. Load `httpd.conf` in your editor. In Fedora and Red Hat Linux, this file is found in `/etc/httpd/conf`.
2. Add the following information to your httpd.conf file, substituting where appropriate with your own local paths.

Note
This should be used instead of the <Directory> block shown above. This should also be above any other `mod_perl` directives within the `httpd.conf` and must be specified in the order as below.

Warning
You should also ensure that you have disabled `KeepAlive` support in your Apache install when utilizing Bugzilla under mod_perl

```
PerlSwitches -w -T
PerlConfigRequire /var/www/html/bugzilla/mod_perl.pl
```

3. `checksetup.pl` can set tighter permissions on Bugzilla's files and directories if it knows what group the web server runs as. Find the `Group` line in `httpd.conf`, place the value found there in the *`$webservergroup`* variable in `localconfig`, then rerun `checksetup.pl`.

On restarting Apache, Bugzilla should now be running within the mod_perl environment. Please ensure you have run checksetup.pl to set permissions before you restart Apache.

Note
Please bear the following points in mind when looking at using Bugzilla under mod_perl:

- mod_perl support in Bugzilla can take up a HUGE amount of RAM. You could be looking at 30MB per httpd child, easily. Basically, you just need a lot of RAM. The more RAM you can get, the better. mod_perl is basically trading RAM for speed. At least 2GB total system RAM is recommended for running Bugzilla under mod_perl.
- Under mod_perl, you have to restart Apache if you make any manual change to any Bugzilla file. You can't just reload--you have to actually *restart* the server (as in make sure it stops and starts again). You *can* change localconfig and the params file manually, if you want, because those are re-read every time you load a page.
- You must run in Apache's Prefork MPM (this is the default). The Worker MPM may not work--we haven't tested Bugzilla's mod_perl support under threads. (And, in fact, we're fairly sure it *won't* work.)
- Bugzilla generally expects to be the only mod_perl application running on your entire server. It may or may not work if there are other applications also running under mod_perl. It does try its best to play nice with other mod_perl applications, but it still may have conflicts.
- It is recommended that you have one Bugzilla instance running under mod_perl on your server. Bugzilla has not been tested with more than one instance running.

Microsoft Internet Information Services

If you are running Bugzilla on Windows and choose to use Microsoft's Internet Information Services or Personal Web Server you will need to perform a number of other configuration steps as explained below. You may also want to refer to the following Microsoft Knowledge Base articles: 245225[5] "HOW TO: Configure and Test a PERL Script with IIS 4.0, 5.0, and 5.1" (for Internet Information Services) and 231998[6] "HOW TO: FP2000: How to Use Perl with Microsoft Personal Web Server on Windows 95/98" (for Personal Web Server).

You will need to create a virtual directory for the Bugzilla install. Put the Bugzilla files in a directory that is named something *other* than what you want your end-users accessing. That is, if you want your users to access your Bugzilla installation through "http://<yourdomainname>/Bugzilla", then do *not* put your Bugzilla files in a directory named "Bugzilla". Instead, place them

[5] http://support.microsoft.com/default.aspx?scid=kb;en-us;245225
[6] http://support.microsoft.com/default.aspx?scid=kb;en-us;231998

in a different location, and then use the IIS Administration tool to create a Virtual Directory named "Bugzilla" that acts as an alias for the actual location of the files. When creating that virtual directory, make sure you add the “Execute (such as ISAPI applications or CGI)” access permission.

You will also need to tell IIS how to handle Bugzilla’s .cgi files. Using the IIS Administration tool again, open up the properties for the new virtual directory and select the Configuration option to access the Script Mappings. Create an entry mapping .cgi to:

```
<full path to perl.exe >\perl.exe -x<full path to Bugzilla> -wT "%s" %s
```

For example:

```
c:\perl\bin\perl.exe -xc:\bugzilla -wT "%s" %s
```

Note
The ActiveState install may have already created an entry for .pl files that is limited to “GET,HEAD,POST”. If so, this mapping should be *removed* as Bugzilla’s .pl files are not designed to be run via a web server.

IIS will also need to know that the index.cgi should be treated as a default document. On the Documents tab page of the virtual directory properties, you need to add index.cgi as a default document type. If you wish, you may remove the other default document types for this particular virtual directory, since Bugzilla doesn’t use any of them.

Also, and this can’t be stressed enough, make sure that files such as `localconfig` and your `data` directory are secured as described in Section 4.2.1.

Bugzilla

Your Bugzilla should now be working. Access `http://<your-bugzilla-server>/` - you should see the Bugzilla front page. If not, consult the Troubleshooting section, Appendix A.

Note
The URL above may be incorrect if you installed Bugzilla into a subdirectory or used a symbolic link from your web site root to the Bugzilla directory.

Log in with the administrator account you defined in the last `checksetup.pl` run. You should go through the Parameters page and see if there are any you wish to change. They key parameters are documented in Section 3.1; you should certainly alter **maintainer** and **urlbase**; you may also want to alter **cookiepath** or **requirelogin**.

Bugzilla has several optional features which require extra configuration. You can read about those in Section 2.3.

Optional Additional Configuration

Bugzilla has a number of optional features. This section describes how to configure or enable them.

Bug Graphs

If you have installed the necessary Perl modules you can start collecting statistics for the nifty Bugzilla graphs.

```
bash# crontab -e
```

This should bring up the crontab file in your editor. Add a cron entry like this to run `collectstats.pl` daily at 5 after midnight:

```
5 0 * * * cd <your-bugzilla-directory> && ./collectstats.pl
```

After two days have passed you'll be able to view bug graphs from the Reports page.

Note
Windows does not have 'cron', but it does have the Task Scheduler, which performs the same duties. There are also third-party tools that can be used to implement cron, such as nncron[a].

[a] http://www.nncron.ru/

The Whining Cron

What good are bugs if they're not annoying? To help make them more so you can set up Bugzilla's automatic whining system to complain at engineers which leave their bugs in the CONFIRMED state without triaging them.

This can be done by adding the following command as a daily crontab entry, in the same manner as explained above for bug graphs. This example runs it at 12.55am.

```
55 0 * * * cd <your-bugzilla-directory> && ./whineatnews.pl
```

Note
Windows does not have 'cron', but it does have the Task Scheduler, which performs the same duties. There are also third-party tools that can be used to implement cron, such as nncron[a].

[a] http://www.nncron.ru/

Whining

As of Bugzilla 2.20, users can configure Bugzilla to regularly annoy them at regular intervals, by having Bugzilla execute saved searches at certain times and emailing the results to the user. This is known as "Whining". The process of configuring Whining is described in Section 5.13, but for it to work a Perl script must be executed at regular intervals.

This can be done by adding the following command as a daily crontab entry, in the same manner as explained above for bug graphs. This example runs it every 15 minutes.

```
*/15 * * * * cd <your-bugzilla-directory> && ./whine.pl
```

Note
Whines can be executed as often as every 15 minutes, so if you specify longer intervals between executions of whine.pl, some users may not be whined at as often as they would expect. Depending on the person, this can either be a very Good Thing or a very Bad Thing.

Note
Windows does not have 'cron', but it does have the Task Scheduler, which performs the same duties. There are also third-party tools that can be used to implement cron, such as nncron[a].

[a] http://www.nncron.ru/

Serving Alternate Formats with the right MIME type

Some Bugzilla pages have alternate formats, other than just plain HTML. In particular, a few Bugzilla pages can output their contents as either XUL (a special Mozilla format, that looks like a program GUI) or RDF (a type of structured XML that can be read by various programs).

In order for your users to see these pages correctly, Apache must send them with the right MIME type. To do this, add the following lines to your Apache configuration, either in the <VirtualHost> section for your Bugzilla, or in the <Directory> section for your Bugzilla:

```
AddType application/vnd.mozilla.xul+xml .xul
AddType application/rdf+xml .rdf
```

Multiple Bugzilla databases with a single installation

The previous instructions referred to a standard installation, with one unique Bugzilla database. However, you may want to host several distinct installations, without having several copies of the code. This is possible by using the PROJECT environment variable. When accessed, Bugzilla checks for the existence of this variable, and if present, uses its value to check for an alternative configuration file named localconfig.<PROJECT> in the same location as the default one (localconfig). It also checks for customized templates in a directory named <PROJECT> in the same location as the default one (template/<langcode>). By default this is template/en/default so PROJECT's templates would be located at template/en/PROJECT.

To set up an alternate installation, just export PROJECT=foo before running **checksetup.pl** for the first time. It will result in a file called localconfig.foo instead of localconfig. Edit this file as described above, with reference to a new database, and re-run **checksetup.pl** to populate it. That's all.

Now you have to configure the web server to pass this environment variable when accessed via an alternate URL, such as virtual host for instance. The following is an example of how you could do it in Apache, other Webservers may differ.

```
<VirtualHost 212.85.153.228:80>
    ServerName foo.bar.baz
    SetEnv PROJECT foo
    Alias /bugzilla /var/www/bugzilla
</VirtualHost>
```

Don't forget to also export this variable before accessing Bugzilla by other means, such as cron tasks for instance.

OS-Specific Installation Notes

Many aspects of the Bugzilla installation can be affected by the operating system you choose to install it on. Sometimes it can be made easier and others more difficult. This section will attempt to help you understand both the difficulties of running on specific operating systems and the utilities available to make it easier.

If you have anything to add or notes for an operating system not covered, please file a bug in Bugzilla Documentation[7].

Microsoft Windows

Making Bugzilla work on Windows is more difficult than making it work on Unix. For that reason, we still recommend doing so on a Unix based system such as GNU/Linux. That said, if you do want to get Bugzilla running on Windows, you will need to make the following adjustments. A detailed step-by-step installation guide for Windows[8] is also available if you need more help with your installation.

[7] http://bugzilla.mozilla.org/enter_bug.cgi?product=Bugzilla;component=Documentation
[8] https://wiki.mozilla.org/Bugzilla:Win32Install

Win32 Perl

Perl for Windows can be obtained from ActiveState[9]. You should be able to find a compiled binary at http://aspn.activestate.com/ASPN/Downloads/ActivePerl/. The following instructions assume that you are using version 5.8.1 of ActiveState.

Note
These instructions are for 32-bit versions of Windows. If you are using a 64-bit version of Windows, you will need to install 32-bit Perl in order to install the 32-bit modules as described below.

Perl Modules on Win32

Bugzilla on Windows requires the same perl modules found in Section 2.1.5. The main difference is that windows uses *PPM* instead of CPAN. ActiveState provides a GUI to manage Perl modules. We highly recommend that you use it. If you prefer to use ppm from the command-line, type:

```
C:\perl> ppm install <module name>
```

Note
The PPM repository stores modules in 'packages' that may have a slightly different name than the module. If retrieving these modules from there, you will need to pay attention to the information provided when you run **checksetup.pl** as it will tell you what package you'll need to install.

Tip
If you are behind a corporate firewall, you will need to let the ActiveState PPM utility know how to get through it to access the repositories by setting the HTTP_proxy system environmental variable. For more information on setting that variable, see the ActiveState documentation.

Serving the web pages

As is the case on Unix based systems, any web server should be able to handle Bugzilla; however, the Bugzilla Team still recommends Apache whenever asked. No matter what web server you choose, be sure to pay attention to the security notes in Section 4.2.1. More information on configuring specific web servers can be found in Section 2.2.4.

Note
The web server looks at `/usr/bin/perl` to call Perl. If you are using Apache on windows, you can set the ScriptInterpreterSource[a] directive in your Apache config file to make it look at the right place: insert the line

```
ScriptInterpreterSource Registry-Strict
```

into your `httpd.conf` file, and create the key

```
HKEY_CLASSES_ROOT\.cgi\Shell\ExecCGI\Command
```

with `C:\Perl\bin\perl.exe -T` as value (adapt to your path if needed) in the registry. When this is done, restart Apache.

[a] http://httpd.apache.org/docs-2.0/mod/core.html#scriptinterpretersource

[9] http://www.activestate.com/

Sending Email

To enable Bugzilla to send email on Windows, the server running the Bugzilla code must be able to connect to, or act as, an SMTP server.

Mac OS X

Making Bugzilla work on Mac OS X requires the following adjustments.

Sendmail

In Mac OS X 10.3 and later, Postfix[10] is used as the built-in email server. Postfix provides an executable that mimics sendmail enough to fool Bugzilla, as long as Bugzilla can find it. Bugzilla is able to find the fake sendmail executable without any assistance.

Libraries & Perl Modules on Mac OS X

Apple does not include the GD library with Mac OS X. Bugzilla needs this for bug graphs.

You can use MacPorts (`http://www.macports.org/`) or Fink (`http://sourceforge.net/projects/fink/`), both of which are similar in nature to the CPAN installer, but install common unix programs.

Follow the instructions for setting up MacPorts or Fink. Once you have one installed, you'll want to use it to install the `gd2` package.

Fink will prompt you for a number of dependencies, type 'y' and hit enter to install all of the dependencies and then watch it work. You will then be able to use *CPAN* to install the GD Perl module.

Note

To prevent creating conflicts with the software that Apple installs by default, Fink creates its own directory tree at `/sw` where it installs most of the software that it installs. This means your libraries and headers will be at `/sw/lib` and `/sw/include` instead of `/usr/lib` and `/usr/include`. When the Perl module config script asks where your `libgd` is, be sure to tell it `/sw/lib`.

Also available via MacPorts and Fink is `expat`. After installing the expat package, you will be able to install XML::Parser using CPAN. If you use fink, there is one caveat. Unlike recent versions of the GD module, XML::Parser doesn't prompt for the location of the required libraries. When using CPAN, you will need to use the following command sequence:

```
# perl -MCPAN -e'look XML::Parser'
# perl Makefile.PL EXPATLIBPATH=/sw/lib EXPATINCPATH=/sw/include
# make; make test; make install
# exit
```

The **look** command will download the module and spawn a new shell with the extracted files as the current working directory.

You should watch the output from these **make** commands, especially "make test" as errors may prevent XML::Parser from functioning correctly with Bugzilla.

The **exit** command will return you to your original shell.

[10] `http://www.postfix.org/`

Linux/BSD Distributions

Many Linux/BSD distributions include Bugzilla and its dependencies in their native package management systems. Installing Bugzilla with root access on any Linux/BSD system should be as simple as finding the Bugzilla package in the package management application and installing it using the normal command syntax. Several distributions also perform the proper web server configuration automatically on installation.

Please consult the documentation of your Linux/BSD distribution for instructions on how to install packages, or for specific instructions on installing Bugzilla with native package management tools. There is also a Bugzilla Wiki Page[11] for distro-specific installation notes.

UNIX (non-root) Installation Notes

Introduction

If you are running a *NIX OS as non-root, either due to lack of access (web hosts, for example) or for security reasons, this will detail how to install Bugzilla on such a setup. It is recommended that you read through the Section 2.1 first to get an idea on the installation steps required. (These notes will reference to steps in that guide.)

MySQL

You may have MySQL installed as root. If you're setting up an account with a web host, a MySQL account needs to be set up for you. From there, you can create the bugs account, or use the account given to you.

Warning
You may have problems trying to set up **GRANT** permissions to the database. If you're using a web host, chances are that you have a separate database which is already locked down (or one big database with limited/no access to the other areas), but you may want to ask your system administrator what the security settings are set to, and/or run the **GRANT** command for you.
Also, you will probably not be able to change the MySQL root user password (for obvious reasons), so skip that step.

Running MySQL as Non-Root

The Custom Configuration Method

Create a file .my.cnf in your home directory (using /home/foo in this example) as follows....

```
[mysqld]
datadir=/home/foo/mymysql
socket=/home/foo/mymysql/thesock
port=8081

[mysql]
socket=/home/foo/mymysql/thesock
port=8081

[mysql.server]
user=mysql
basedir=/var/lib

[safe_mysqld]
err-log=/home/foo/mymysql/the.log
pid-file=/home/foo/mymysql/the.pid
```

[11] https://wiki.mozilla.org/Bugzilla:Prerequisites

The Custom Built Method

You can install MySQL as a not-root, if you really need to. Build it with PREFIX set to `/home/foo/mysql`, or use pre-installed executables, specifying that you want to put all of the data files in `/home/foo/mysql/data`. If there is another MySQL server running on the system that you do not own, use the -P option to specify a TCP port that is not in use.

Starting the Server

After your mysqld program is built and any .my.cnf file is in place, you must initialize the databases (ONCE).

```
bash$ mysql_install_db
```

Then start the daemon with

```
bash$ safe_mysql &
```

After you start mysqld the first time, you then connect to it as "root" and **GRANT** permissions to other users. (Again, the MySQL root account has nothing to do with the *NIX root account.)

Note
You will need to start the daemons yourself. You can either ask your system administrator to add them to system startup files, or add a crontab entry that runs a script to check on these daemons and restart them if needed.

Warning
Do NOT run daemons or other services on a server without first consulting your system administrator! Daemons use up system resources and running one may be in violation of your terms of service for any machine on which you are a user!

Perl

On the extremely rare chance that you don't have Perl on the machine, you will have to build the sources yourself. The following commands should get your system installed with your own personal version of Perl:

```
bash$ wget http://perl.org/CPAN/src/stable.tar.gz
bash$ tar zvxf stable.tar.gz
bash$ cd perl-5.8.1
bash$ sh Configure -de -Dprefix=/home/foo/perl
bash$ make && make test && make install
```

Once you have Perl installed into a directory (probably in `~/perl/bin`), you will need to install the Perl Modules, described below.

Perl Modules

Installing the Perl modules as a non-root user is accomplished by running the `install-module.pl` script. For more details on this script, see `install-module.pl` documentation[12]

HTTP Server

Ideally, this also needs to be installed as root and run under a special web server account. As long as the web server will allow the running of *.cgi files outside of a cgi-bin, and a way of denying web access to certain files (such as a .htaccess file), you should be good in this department.

[12] `../html/api/install-module.html`

Running Apache as Non-Root

You can run Apache as a non-root user, but the port will need to be set to one above 1024. If you type **httpd -V**, you will get a list of the variables that your system copy of httpd uses. One of those, namely HTTPD_ROOT, tells you where that installation looks for its config information.

From there, you can copy the config files to your own home directory to start editing. When you edit those and then use the -d option to override the HTTPD_ROOT compiled into the web server, you get control of your own customized web server.

Note
You will need to start the daemons yourself. You can either ask your system administrator to add them to system startup files, or add a crontab entry that runs a script to check on these daemons and restart them if needed.

Warning
Do NOT run daemons or other services on a server without first consulting your system administrator! Daemons use up system resources and running one may be in violation of your terms of service for any machine on which you are a user!

Bugzilla

When you run **./checksetup.pl** to create the `localconfig` file, it will list the Perl modules it finds. If one is missing, go back and double-check the module installation from Section 2.6.4, then delete the `localconfig` file and try again.

Warning
One option in `localconfig` you might have problems with is the web server group. If you can't successfully browse to the `index.cgi` (like a Forbidden error), you may have to relax your permissions, and blank out the web server group. Of course, this may pose as a security risk. Having a properly jailed shell and/or limited access to shell accounts may lessen the security risk, but use at your own risk.

suexec or shared hosting

If you are running on a system that uses suexec (most shared hosting environments do this), you will need to set the *webserver-group* value in `localconfig` to match *your* primary group, rather than the one the web server runs under. You will need to run the following shell commands after running **./checksetup.pl**, every time you run it (or modify `checksetup.pl` to do them for you via the system() command).

```
for i in docs graphs images js skins; do find $i -type d -exec chmod o+rx {} \; ; done
for i in jpg gif css js png html rdf xul; do find . -name \*.$i -exec chmod o+r {} \; ;  ↩
    done
find . -name .htaccess -exec chmod o+r {} \;
chmod o+x . data data/webdot
```

Pay particular attention to the number of semicolons and dots. They are all important. A future version of Bugzilla will hopefully be able to do this for you out of the box.

Upgrading to New Releases

Upgrading to new Bugzilla releases is very simple. There is a script named `checksetup.pl` included with Bugzilla that will automatically do all of the database migration for you.

The following sections explain how to upgrade from one version of Bugzilla to another. Whether you are upgrading from one bug-fix version to another (such as 4.2 to 4.2.1) or from one major version to another (such as from 4.0 to 4.2), the instructions are always the same.

Note
Any examples in the following sections are written as though the user were updating to version 4.2.1, but the procedures are the same no matter what version you're updating to. Also, in the examples, the user's Bugzilla installation is found at `/var/www/html/bugzilla`. If that is not the same as the location of your Bugzilla installation, simply substitute the proper paths where appropriate.

Before You Upgrade

Before you start your upgrade, there are a few important steps to take:

1. Read the Release Notes[13] of the version you're upgrading to, particularly the "Notes for Upgraders" section.
2. View the Sanity Check (Section 3.16) page on your installation before upgrading. Attempt to fix all warnings that the page produces before you go any further, or you may experience problems during your upgrade.
3. Shut down your Bugzilla installation by putting some HTML or text in the shutdownhtml parameter (see Section 3.1).
4. Make a backup of the Bugzilla database. *THIS IS VERY IMPORTANT*. If anything goes wrong during the upgrade, your installation can be corrupted beyond recovery. Having a backup keeps you safe.

Warning
Upgrading is a one-way process. You cannot "downgrade" an upgraded Bugzilla. If you wish to revert to the old Bugzilla version for any reason, you will have to restore your database from this backup.

Here are some sample commands you could use to backup your database, depending on what database system you're using. You may have to modify these commands for your particular setup.

MySQL: mysqldump --opt -u bugs -p bugs > bugs.sql

PostgreSQL: pg_dump --no-privileges --no-owner -h localhost -U bugs > bugs.sql

Getting The New Bugzilla

There are three ways to get the new version of Bugzilla. We'll list them here briefly and then explain them more later.

Bzr (Section 2.7.2.2) If you have **bzr** installed on your machine and you have Internet access, this is the easiest way to upgrade, particularly if you have made modifications to the code or templates of Bugzilla.

Download the tarball (Section 2.7.2.3) This is a very simple way to upgrade, and good if you haven't made many (or any) modifications to the code or templates of your Bugzilla.

Patches (Section 2.7.2.4) If you have made modifications to your Bugzilla, and you don't have Internet access or you don't want to use bzr, then this is the best way to upgrade.

You can only do minor upgrades (such as 4.2 to 4.2.1 or 4.2.1 to 4.2.2) with patches.

If you have modified your Bugzilla

If you have modified the code or templates of your Bugzilla, then upgrading requires a bit more thought and effort. A discussion of the various methods of updating compared with degree and methods of local customization can be found in Section 6.3.2.

The larger the jump you are trying to make, the more difficult it is going to be to upgrade if you have made local customizations. Upgrading from 4.2 to 4.2.1 should be fairly painless even if you are heavily customized, but going from 2.18 to 4.2 is going to mean a fair bit of work re-writing your local changes to use the new files, logic, templates, etc. If you have done no local changes at all, however, then upgrading should be approximately the same amount of work regardless of how long it has been since your version was released.

[13] http://www.bugzilla.org/releases/

Upgrading using Bzr

This requires that you have bzr installed (most Unix machines do), and requires that you are able to access bzr.mozilla.org[14], which may not be an option if you don't have Internet access.

The following shows the sequence of commands needed to update a Bugzilla installation via Bzr, and a typical series of results. These commands assume that you already have Bugzilla installed using Bzr.

Warning
If your installation is still using CVS, you must first convert it to Bzr. A very detailed step by step documentation can be found on wiki.mozilla.org[a].

[a] https://wiki.mozilla.org/Bugzilla:Moving_From_CVS_To_Bazaar

```
bash$ cd /var/www/html/bugzilla
bash$ bzr switch 4.2 (only run this command when not yet running 4.2)
bash$ bzr up -r tag:bugzilla-4.2.1
+N  extensions/MoreBugUrl/
+N  extensions/MoreBugUrl/Config.pm
+N  extensions/MoreBugUrl/Extension.pm
...
 M  Bugzilla/Attachment.pm
 M  Bugzilla/Attachment/PatchReader.pm
 M  Bugzilla/Bug.pm
...
All changes applied successfully.
```

Caution
If a line in the output from **bzr up** mentions a conflict, then that represents a file with local changes that Bzr was unable to properly merge. You need to resolve these conflicts manually before Bugzilla (or at least the portion using that file) will be usable.

Upgrading using the tarball

If you are unable (or unwilling) to use Bzr, another option that's always available is to obtain the latest tarball from the Download Page[15] and create a new Bugzilla installation from that.

This sequence of commands shows how to get the tarball from the command-line; it is also possible to download it from the site directly in a web browser. If you go that route, save the file to the `/var/www/html` directory (or its equivalent, if you use something else) and omit the first three lines of the example.

```
bash$ cd /var/www/html
bash$ wget http://ftp.mozilla.org/pub/mozilla.org/webtools/bugzilla-4.2.1.tar.gz
(Output omitted)
bash$ tar xzvf bugzilla-4.2.1.tar.gz
bugzilla-4.2.1/
bugzilla-4.2.1/colchange.cgi
(Output truncated)
bash$ cd bugzilla-4.2.1
bash$ cp ../bugzilla/localconfig* .
bash$ cp -r ../bugzilla/data .
bash$ cd ..
bash$ mv bugzilla bugzilla.old
bash$ mv bugzilla-4.2.1 bugzilla
```

[14] http://bzr.mozilla.org/bugzilla/
[15] http://www.bugzilla.org/download/

Warning
The **cp** commands both end with periods which is a very important detail--it means that the destination directory is the current working directory.

Caution
If you have some extensions installed, you will have to copy them to the new bugzilla directory too. Extensions are located in `bugzilla/extensions/`. Only copy those you installed, not those managed by the Bugzilla team.

This upgrade method will give you a clean install of Bugzilla. That's fine if you don't have any local customizations that you want to maintain. If you do have customizations, then you will need to reapply them by hand to the appropriate files.

Upgrading using patches

A patch is a collection of all the bug fixes that have been made since the last bug-fix release.

If you are doing a bug-fix upgrade—that is, one where only the last number of the revision changes, such as from 4.2 to 4.2.1—then you have the option of obtaining and applying a patch file from the Download Page[16].

As above, this example starts with obtaining the file via the command line. If you have already downloaded it, you can omit the first two commands.

```
bash$ cd /var/www/html/bugzilla
bash$ wget http://ftp.mozilla.org/pub/mozilla.org/webtools/bugzilla-4.2-to-4.2.1.diff.gz
(Output omitted)
bash$ gunzip bugzilla-4.2-to-4.2.1.diff.gz
bash$ patch -p1 < bugzilla-4.2-to-4.2.1.diff
patching file Bugzilla/Constants.pm
patching file enter_bug.cgi
(etc.)
```

Warning
Be aware that upgrading from a patch file does not change the entries in your `.bzr` directory. This could make it more difficult to upgrade using Bzr (Section 2.7.2.2) in the future.

Completing Your Upgrade

Now that you have the new Bugzilla code, there are a few final steps to complete your upgrade.

1. If your new Bugzilla installation is in a different directory or on a different machine than your old Bugzilla installation, make sure that you have copied the `data` directory and the `localconfig` file from your old Bugzilla installation. (If you followed the tarball instructions above, this has already happened.)
2. If this is a major update, check that the configuration (Section 2.2) for your new Bugzilla is up-to-date. Sometimes the configuration requirements change between major versions.
3. If you didn't do it as part of the above configuration step, now you need to run **checksetup.pl**, which will do everything required to convert your existing database and settings for the new version:

[16] http://www.bugzilla.org/download/

```
bash$ cd /var/www/html/bugzilla
bash$ ./checksetup.pl
```

Warning
The period at the beginning of the command **./checksetup.pl** is important and cannot be omitted.

Caution
If this is a major upgrade (say, 3.6 to 4.2 or similar), running **checksetup.pl** on a large installation (75,000 or more bugs) can take a long time, possibly several hours.

4. Clear any HTML or text that you put into the shutdownhtml parameter, to re-activate Bugzilla.
5. View the Sanity Check (Section 3.16) page in your upgraded Bugzilla.

   It is recommended that, if possible, you fix any problems you see, immediately. Failure to do this may mean that Bugzilla will not work correctly. Be aware that if the sanity check page contains more errors after an upgrade, it doesn't necessarily mean there are more errors in your database than there were before, as additional tests are added to the sanity check over time, and it is possible that those errors weren't being checked for in the old version.

Automatic Notifications of New Releases

Bugzilla 3.0 introduced the ability to automatically notify administrators when new releases are available, based on the `upgrade_notification` parameter, see Section 3.1. Administrators will see these notifications when they access the `index.cgi` page, i.e. generally when logging in. Bugzilla will check once per day for new releases, unless the parameter is set to "disabled". If you are behind a proxy, you may have to set the `proxy_url` parameter accordingly. If the proxy requires authentication, use the `http://user:pass@proxy_url/` syntax.

Chapter 3

Administering Bugzilla

Bugzilla Configuration

Bugzilla is configured by changing various parameters, accessed from the "Parameters" link in the Administration page (the Administration page can be found by clicking the "Administration" link in the footer). The parameters are divided into several categories, accessed via the menu on the left. Following is a description of the different categories and important parameters within those categories.

Required Settings

The core required parameters for any Bugzilla installation are set here. These parameters must be set before a new Bugzilla installation can be used. Administrators should review this list before deploying a new Bugzilla installation.

maintainer Email address of the person responsible for maintaining this Bugzilla installation. The address need not be that of a valid Bugzilla account.

urlbase Defines the fully qualified domain name and web server path to this Bugzilla installation.

For example, if the Bugzilla query page is `http://www.foo.com/bugzilla/query.cgi`, the “urlbase” should be set to `http://www.foo.com/bugzilla/`.

docs_urlbase Defines path to the Bugzilla documentation. This can be a fully qualified domain name, or a path relative to "urlbase".

For example, if the "Bugzilla Configuration" page of the documentation is `http://www.foo.com/bugzilla/docs/html/parameters.html`, set the “docs_urlbase” to `http://www.foo.com/bugzilla/docs/html/`.

sslbase Defines the fully qualified domain name and web server path for HTTPS (SSL) connections to this Bugzilla installation.

For example, if the Bugzilla main page is `https://www.foo.com/bugzilla/index.cgi`, the “sslbase” should be set to `https://www.foo.com/bugzilla/`.

ssl_redirect If enabled, Bugzilla will force HTTPS (SSL) connections, by automatically redirecting any users who try to use a non-SSL connection.

cookiedomain Defines the domain for Bugzilla cookies. This is typically left blank. If there are multiple hostnames that point to the same webserver, which require the same cookie, then this parameter can be utilized. For example, If your website is at `https://www.foo.com/`, setting this to `.foo.com/` will also allow `bar.foo.com/` to access Bugzilla cookies.

cookiepath Defines a path, relative to the web server root, that Bugzilla cookies will be restricted to. For example, if the **urlbase** is set to `http://www.foo.com/bugzilla/`, the **cookiepath** should be set to `/bugzilla/`. Setting it to "/" will allow all sites served by this web server or virtual host to read Bugzilla cookies.

utf8 Determines whether to use UTF-8 (Unicode) encoding for all text in Bugzilla. New installations should set this to true to avoid character encoding problems. Existing databases should set this to true only after the data has been converted from existing legacy character encoding to UTF-8, using the `contrib/recode.pl` script.

Note
If you turn this parameter from "off" to "on", you must re-run `checksetup.pl` immediately afterward.

---

shutdownhtml If there is any text in this field, this Bugzilla installation will be completely disabled and this text will appear instead of all Bugzilla pages for all users, including Admins. Used in the event of site maintenance or outage situations.

Note
Although regular log-in capability is disabled while **shutdownhtml** is enabled, safeguards are in place to protect the unfortunate admin who loses connection to Bugzilla. Should this happen to you, go directly to the `editparams.cgi` (by typing the URL in manually, if necessary). Doing this will prompt you to log in, and your name/password will be accepted here (but nowhere else).

---

announcehtml Any text in this field will be displayed at the top of every HTML page in this Bugzilla installation. The text is not wrapped in any tags. For best results, wrap the text in a " " tag. Any style attributes from the CSS can be applied. For example, to make the text green inside of a red box, add "id=message" to the " " tag.

proxy_url If this Bugzilla installation is behind a proxy, enter the proxy information here to enable Bugzilla to access the Internet. Bugzilla requires Internet access to utilize the **upgrade_notification** parameter (below). If the proxy requires authentication, use the syntax: `http://user:pass@proxy_url/`.

upgrade_notification Enable or disable a notification on the homepage of this Bugzilla installation when a newer version of Bugzilla is available. This notification is only visible to administrators. Choose "disabled", to turn off the notification. Otherwise, choose which version of Bugzilla you want to be notified about: "development_snapshot" is the latest release on the trunk; "latest_stable_release" is the most recent release available on the most recent stable branch; "stable_branch_release" the most recent release on the branch this installation is based on.

Administrative Policies

This page contains parameters for basic administrative functions. Options include whether to allow the deletion of bugs and users, and whether to allow users to change their email address.

User Authentication

This page contains the settings that control how this Bugzilla installation will do its authentication. Choose what authentication mechanism to use (the Bugzilla database, or an external source such as LDAP), and set basic behavioral parameters. For example, choose whether to require users to login to browse bugs, the management of authentication cookies, and the regular expression used to validate email addresses. Some parameters are highlighted below.

emailregexp Defines the regular expression used to validate email addresses used for login names. The default attempts to match fully qualified email addresses (i.e. 'user@example.com') in a slightly more restrictive way than what is allowed in RFC 2822. Some Bugzilla installations allow only local user names (i.e 'user' instead of 'user@example.com'). In that case, this parameter should be used to define the email domain.

emailsuffix This string is appended to login names when actually sending email to a user. For example, If **emailregexp** has been set to allow local usernames, then this parameter would contain the email domain for all users (i.e. '@example.com').

Attachments

This page allows for setting restrictions and other parameters regarding attachments to bugs. For example, control size limitations and whether to allow pointing to external files via a URI.

Bug Change Policies

Set policy on default behavior for bug change events. For example, choose which status to set a bug to when it is marked as a duplicate, and choose whether to allow bug reporters to set the priority or target milestone. Also allows for configuration of what changes should require the user to make a comment, described below.

commenton* All these fields allow you to dictate what changes can pass without comment, and which must have a comment from the person who changed them. Often, administrators will allow users to add themselves to the CC list, accept bugs, or change the Status Whiteboard without adding a comment as to their reasons for the change, yet require that most other changes come with an explanation.

Set the "commenton" options according to your site policy. It is a wise idea to require comments when users resolve, reassign, or reopen bugs at the very least.

Note
It is generally far better to require a developer comment when resolving bugs than not. Few things are more annoying to bug database users than having a developer mark a bug "fixed" without any comment as to what the fix was (or even that it was truly fixed!)

---

noresolveonopenblockers This option will prevent users from resolving bugs as FIXED if they have unresolved dependencies. Only the FIXED resolution is affected. Users will be still able to resolve bugs to resolutions other than FIXED if they have unresolved dependent bugs.

Bug Fields

The parameters in this section determine the default settings of several Bugzilla fields for new bugs, and also control whether certain fields are used. For example, choose whether to use the "target milestone" field or the "status whiteboard" field.

useqacontact This allows you to define an email address for each component, in addition to that of the default assignee, who will be sent carbon copies of incoming bugs.

usestatuswhiteboard This defines whether you wish to have a free-form, overwritable field associated with each bug. The advantage of the Status Whiteboard is that it can be deleted or modified with ease, and provides an easily-searchable field for indexing some bugs that have some trait in common.

Bug Moving

This page controls whether this Bugzilla installation allows certain users to move bugs to an external database. If bug moving is enabled, there are a number of parameters that control bug moving behaviors. For example, choose which users are allowed to move bugs, the location of the external database, and the default product and component that bugs moved *from* other bug databases to this Bugzilla installation are assigned to.

Dependency Graphs

This page has one parameter that sets the location of a Web Dot server, or of the Web Dot binary on the local system, that is used to generate dependency graphs. Web Dot is a CGI program that creates images from `.dot` graphic description files. If no Web Dot server or binary is specified, then dependency graphs will be disabled.

Group Security

Bugzilla allows for the creation of different groups, with the ability to restrict the visibility of bugs in a group to a set of specific users. Specific products can also be associated with groups, and users restricted to only see products in their groups. Several parameters are described in more detail below. Most of the configuration of groups and their relationship to products is done on the "Groups" and "Product" pages of the "Administration" area. The options on this page control global default behavior. For more information on Groups and Group Security, see Section 3.15

makeproductgroups Determines whether or not to automatically create groups when new products are created. If this is on, the groups will be used for querying bugs.

usevisibilitygroups If selected, user visibility will be restricted to members of groups, as selected in the group configuration settings. Each user-defined group can be allowed to see members of selected other groups. For details on configuring groups (including the visibility restrictions) see Section 3.15.2.

querysharegroup The name of the group of users who are allowed to share saved searches with one another. For more information on using saved searches, see Saved Searches.

LDAP Authentication

LDAP authentication is a module for Bugzilla's plugin authentication architecture. This page contains all the parameters necessary to configure Bugzilla for use with LDAP authentication.

The existing authentication scheme for Bugzilla uses email addresses as the primary user ID, and a password to authenticate that user. All places within Bugzilla that require a user ID (e.g assigning a bug) use the email address. The LDAP authentication builds on top of this scheme, rather than replacing it. The initial log-in is done with a username and password for the LDAP directory. Bugzilla tries to bind to LDAP using those credentials and, if successful, tries to map this account to a Bugzilla account. If an LDAP mail attribute is defined, the value of this attribute is used, otherwise the "emailsuffix" parameter is appended to LDAP username to form a full email address. If an account for this address already exists in the Bugzilla installation, it will log in to that account. If no account for that email address exists, one is created at the time of login. (In this case, Bugzilla will attempt to use the "displayName" or "cn" attribute to determine the user's full name.) After authentication, all other user-related tasks are still handled by email address, not LDAP username. For example, bugs are still assigned by email address and users are still queried by email address.

Caution

Because the Bugzilla account is not created until the first time a user logs in, a user who has not yet logged is unknown to Bugzilla. This means they cannot be used as an assignee or QA contact (default or otherwise), added to any CC list, or any other such operation. One possible workaround is the `bugzilla_ldapsync.rb` script in the *contrib* directory. Another possible solution is fixing bug 201069[a].

[a] https://bugzilla.mozilla.org/show_bug.cgi?id=201069

Parameters required to use LDAP Authentication:

user_verify_class If you want to list "LDAP" here, make sure to have set up the other parameters listed below. Unless you have other (working) authentication methods listed as well, you may otherwise not be able to log back in to Bugzilla once you log out. If this happens to you, you will need to manually edit `data/params` and set user_verify_class to "DB".

LDAPserver This parameter should be set to the name (and optionally the port) of your LDAP server. If no port is specified, it assumes the default LDAP port of 389.

For example: "ldap.company.com" or "ldap.company.com:3268"

You can also specify a LDAP URI, so as to use other protocols, such as LDAPS or LDAPI. If port was not specified in the URI, the default is either 389 or 636 for 'LDAP' and 'LDAPS' schemes respectively.

Tip

In order to use SSL with LDAP, specify a URI with "ldaps://". This will force the use of SSL over port 636.

For example, normal LDAP: "ldap://ldap.company.com", LDAP over SSL: "ldaps://ldap.company.com" or LDAP over a UNIX domain socket "ldapi://%2fvar%2flib%2fldap_sock".

LDAPbinddn [Optional] Some LDAP servers will not allow an anonymous bind to search the directory. If this is the case with your configuration you should set the LDAPbinddn parameter to the user account Bugzilla should use instead of the anonymous bind.

Ex. "cn=default,cn=user:password"

LDAPBaseDN The LDAPBaseDN parameter should be set to the location in your LDAP tree that you would like to search for email addresses. Your uids should be unique under the DN specified here.

Ex. “ou=People,o=Company”

LDAPuidattribute The LDAPuidattribute parameter should be set to the attribute which contains the unique UID of your users. The value retrieved from this attribute will be used when attempting to bind as the user to confirm their password.

Ex. “uid”

LDAPmailattribute The LDAPmailattribute parameter should be the name of the attribute which contains the email address your users will enter into the Bugzilla login boxes.

Ex. “mail”

RADIUS Authentication

RADIUS authentication is a module for Bugzilla’s plugin authentication architecture. This page contains all the parameters necessary for configuring Bugzilla to use RADIUS authentication.

Note
Most caveats that apply to LDAP authentication apply to RADIUS authentication as well. See Section 3.1.10 for details.

Parameters required to use RADIUS Authentication:

user_verify_class If you want to list “RADIUS” here, make sure to have set up the other parameters listed below. Unless you have other (working) authentication methods listed as well, you may otherwise not be able to log back in to Bugzilla once you log out. If this happens to you, you will need to manually edit `data/params` and set user_verify_class to “DB”.

RADIUS_server This parameter should be set to the name (and optionally the port) of your RADIUS server.

RADIUS_secret This parameter should be set to the RADIUS server’s secret.

RADIUS_email_suffix Bugzilla needs an e-mail address for each user account. Therefore, it needs to determine the e-mail address corresponding to a RADIUS user. Bugzilla offers only a simple way to do this: it can concatenate a suffix to the RADIUS user name to convert it into an e-mail address. You can specify this suffix in the RADIUS_email_suffix parameter.

If this simple solution does not work for you, you’ll probably need to modify `Bugzilla/Auth/Verify/RADIUS.pm` to match your requirements.

Email

This page contains all of the parameters for configuring how Bugzilla deals with the email notifications it sends. See below for a summary of important options.

mail_delivery_method This is used to specify how email is sent, or if it is sent at all. There are several options included for different MTAs, along with two additional options that disable email sending. "Test" does not send mail, but instead saves it in `data/mailer.testfile` for later review. "None" disables email sending entirely.

mailfrom This is the email address that will appear in the "From" field of all emails sent by this Bugzilla installation. Some email servers require mail to be from a valid email address, therefore it is recommended to choose a valid email address here.

smtpserver This is the SMTP server address, if the “mail_delivery_method” parameter is set to SMTP. Use "localhost" if you have a local MTA running, otherwise use a remote SMTP server. Append ":" and the port number, if a non-default port is needed.

smtp_username Username to use for SASL authentication to the SMTP server. Leave this parameter empty if your server does not require authentication.

smtp_password Password to use for SASL authentication to the SMTP server. This parameter will be ignored if the “smtp_username” parameter is left empty.

smtp_ssl Enable SSL support for connection to the SMTP server.

smtp_debug This parameter allows you to enable detailed debugging output. Log messages are printed the web server’s error log.

whinedays Set this to the number of days you want to let bugs go in the CONFIRMED state before notifying people they have untouched new bugs. If you do not plan to use this feature, simply do not set up the whining cron job described in the installation instructions, or set this value to "0" (never whine).

globalwatcher This allows you to define specific users who will receive notification each time a new bug in entered, or when an existing bug changes, according to the normal groupset permissions. It may be useful for sending notifications to a mailing-list, for instance.

Patch Viewer

This page contains configuration parameters for the CVS server, Bonsai server and LXR server that Bugzilla will use to enable the features of the Patch Viewer. Bonsai is a tool that enables queries to a CVS tree. LXR is a tool that can cross reference and index source code.

Query Defaults

This page controls the default behavior of Bugzilla in regards to several aspects of querying bugs. Options include what the default query options are, what the "My Bugs" page returns, whether users can freely add bugs to the quip list, and how many duplicate bugs are needed to add a bug to the "most frequently reported" list.

Shadow Database

This page controls whether a shadow database is used, and all the parameters associated with the shadow database. Versions of Bugzilla prior to 3.2 used the MyISAM table type, which supports only table-level write locking. With MyISAM, any time someone is making a change to a bug, the entire table is locked until the write operation is complete. Locking for write also blocks reads until the write is complete.

The “shadowdb” parameter was designed to get around this limitation. While only a single user is allowed to write to a table at a time, reads can continue unimpeded on a read-only shadow copy of the database.

Note
As of version 3.2, Bugzilla no longer uses the MyISAM table type. Instead, InnoDB is used, which can do transaction-based locking. Therefore, the limitations the Shadow Database feature was designed to workaround no longer exist.

User Matching

The settings on this page control how users are selected and queried when adding a user to a bug. For example, users need to be selected when choosing who the bug is assigned to, adding to the CC list or selecting a QA contact. With the "usemenuforusers" parameter, it is possible to configure Bugzilla to display a list of users in the fields instead of an empty text field. This should only be used in Bugzilla installations with a small number of users. If users are selected via a text box, this page also contains parameters for how user names can be queried and matched when entered.

Another setting called ’ajax_user_autocompletion’ enables certain user fields to display a list of matched user names as a drop down after typing a few characters. Note that it is recommended to use mod_perl when enabling ’ajax_user_autocompletion’.

User Administration

Creating the Default User

When you first run checksetup.pl after installing Bugzilla, it will prompt you for the administrative username (email address) and password for this "super user". If for some reason you delete the "super user" account, re-running checksetup.pl will again prompt you for this username and password.

Tip
If you wish to add more administrative users, add them to the "admin" group and, optionally, edit the tweakparams, editusers, creategroups, editcomponents, and editkeywords groups to add the entire admin group to those groups (which is the case by default).

Managing Other Users

Searching for existing users

If you have “editusers” privileges or if you are allowed to grant privileges for some groups, the “Users” link will appear in the Administration page.

The first screen is a search form to search for existing user accounts. You can run searches based either on the user ID, real name or login name (i.e. the email address, or just the first part of the email address if the "emailsuffix" parameter is set). The search can be conducted in different ways using the listbox to the right of the text entry box. You can match by case-insensitive substring (the default), regular expression, a *reverse* regular expression match (which finds every user name which does NOT match the regular expression), or the exact string if you know exactly who you are looking for. The search can be restricted to users who are in a specific group. By default, the restriction is turned off.

The search returns a list of users matching your criteria. User properties can be edited by clicking the login name. The Account History of a user can be viewed by clicking the "View" link in the Account History column. The Account History displays changes that have been made to the user account, the time of the change and the user who made the change. For example, the Account History page will display details of when a user was added or removed from a group.

Creating new users

Self-registration

By default, users can create their own user accounts by clicking the “New Account” link at the bottom of each page (assuming they aren’t logged in as someone else already). If you want to disable this self-registration, or if you want to restrict who can create his own user account, you have to edit the “createemailregexp” parameter in the “Configuration” page, see Section 3.1.

Accounts created by an administrator

Users with “editusers” privileges, such as administrators, can create user accounts for other users:

1. After logging in, click the "Users" link at the footer of the query page, and then click "Add a new user".
2. Fill out the form presented. This page is self-explanatory. When done, click "Submit".

Note
Adding a user this way will *not* send an email informing them of their username and password. While useful for creating dummy accounts (watchers which shuttle mail to another system, for instance, or email addresses which are a mailing list), in general it is preferable to log out and use the “New Account” button to create users, as it will pre-populate all the required fields and also notify the user of her account name and password.

Modifying Users

Once you have found your user, you can change the following fields:

- *Login Name*: This is generally the user's full email address. However, if you are using the "emailsuffix" parameter, this may just be the user's login name. Note that users can now change their login names themselves (to any valid email address).
- *Real Name*: The user's real name. Note that Bugzilla does not require this to create an account.
- *Password*: You can change the user's password here. Users can automatically request a new password, so you shouldn't need to do this often. If you want to disable an account, see Disable Text below.
- *Bugmail Disabled*: Mark this checkbox to disable bugmail and whinemail completely for this account. This checkbox replaces the data/nomail file which existed in older versions of Bugzilla.
- *Disable Text*: If you type anything in this box, including just a space, the user is prevented from logging in, or making any changes to bugs via the web interface. The HTML you type in this box is presented to the user when they attempt to perform these actions, and should explain why the account was disabled.

  Users with disabled accounts will continue to receive mail from Bugzilla; furthermore, they will not be able to log in themselves to change their own preferences and stop it. If you want an account (disabled or active) to stop receiving mail, simply check the "Bugmail Disabled" checkbox above.

  Note
  Even users whose accounts have been disabled can still submit bugs via the e-mail gateway, if one exists. The e-mail gateway should *not* be enabled for secure installations of Bugzilla.

  

  Warning
  Don't disable all the administrator accounts!

- <*groupname*>: If you have created some groups, e.g. "securitysensitive", then checkboxes will appear here to allow you to add users to, or remove them from, these groups. The first checkbox gives the user the ability to add and remove other users as members of this group. The second checkbox adds the user himself as a member of the group.
- *canconfirm*: This field is only used if you have enabled the "unconfirmed" status. If you enable this for a user, that user can then move bugs from "Unconfirmed" to a "Confirmed" status (e.g.: "New" status).
- *creategroups*: This option will allow a user to create and destroy groups in Bugzilla.
- *editbugs*: Unless a user has this bit set, they can only edit those bugs for which they are the assignee or the reporter. Even if this option is unchecked, users can still add comments to bugs.
- *editcomponents*: This flag allows a user to create new products and components, as well as modify and destroy those that have no bugs associated with them. If a product or component has bugs associated with it, those bugs must be moved to a different product or component before Bugzilla will allow them to be destroyed.
- *editkeywords*: If you use Bugzilla's keyword functionality, enabling this feature allows a user to create and destroy keywords. As always, the keywords for existing bugs containing the keyword the user wishes to destroy must be changed before Bugzilla will allow it to die.
- *editusers*: This flag allows a user to do what you're doing right now: edit other users. This will allow those with the right to do so to remove administrator privileges from other users or grant them to themselves. Enable with care.
- *tweakparams*: This flag allows a user to change Bugzilla's Params (using `editparams.cgi`.)
- <*productname*>: This allows an administrator to specify the products in which a user can see bugs. If you turn on the "makeproductgroups" parameter in the Group Security Panel in the Parameters page, then Bugzilla creates one group per product (at the time you create the product), and this group has exactly the same name as the product itself. Note that for products that already exist when the parameter is turned on, the corresponding group will not be created. The user must still have the "editbugs" privilege to edit bugs in these products.

Deleting Users

If the “allowuserdeletion” parameter is turned on, see Section 3.1, then you can also delete user accounts. Note that this is most of the time not the best thing to do. If only a warning in a yellow box is displayed, then the deletion is safe. If a warning is also displayed in a red box, then you should NOT try to delete the user account, else you will get referential integrity problems in your database, which can lead to unexpected behavior, such as bugs not appearing in bug lists anymore, or data displaying incorrectly. You have been warned!

Impersonating Users

There may be times when an administrator would like to do something as another user. The **sudo** feature may be used to do this.

Note
To use the sudo feature, you must be in the *bz_sudoers* group. By default, all administrators are in this group.

If you have access to this feature, you may start a session by going to the Edit Users page, Searching for a user and clicking on their login. You should see a link below their login name titled "Impersonate this user". Click on the link. This will take you to a page where you will see a description of the feature and instructions for using it. After reading the text, simply enter the login of the user you would like to impersonate, provide a short message explaining why you are doing this, and press the button.

As long as you are using this feature, everything you do will be done as if you were logged in as the user you are impersonating.

Warning
The user you are impersonating will not be told about what you are doing. If you do anything that results in mail being sent, that mail will appear to be from the user you are impersonating. You should be extremely careful while using this feature.

Classifications

Classifications tend to be used in order to group several related products into one distinct entity.

The classifications layer is disabled by default; it can be turned on or off using the useclassification parameter, in the *Bug Fields* section of the edit parameters screen.

Access to the administration of classifications is controlled using the *editclassifications* system group, which defines a privilege for creating, destroying, and editing classifications.

When activated, classifications will introduce an additional step when filling bugs (dedicated to classification selection), and they will also appear in the advanced search form.

Products

Products typically represent real-world shipping products. Products can be given Classifications. For example, if a company makes computer games, they could have a classification of "Games", and a separate product for each game. This company might also have a “Common” product for units of technology used in multiple games, and perhaps a few special products that represent items that are not actually shipping products (for example, "Website", or "Administration").

Many of Bugzilla’s settings are configurable on a per-product basis. The number of “votes” available to users is set per-product, as is the number of votes required to move a bug automatically from the UNCONFIRMED status to the CONFIRMED status.

When creating or editing products the following options are available:

Product The name of the product

Description A brief description of the product

Default milestone Select the default milestone for this product.

Closed for bug entry Select this box to prevent new bugs from being entered against this product.

Maximum votes per person Maximum votes a user is allowed to give for this product

Maximum votes a person can put on a single bug Maximum votes a user is allowed to give for this product in a single bug

Confirmation threshold Number of votes needed to automatically remove any bug against this product from the UNCONFIRMED state

Version Specify which version of the product bugs will be entered against.

Create chart datasets for this product Select to make chart datasets available for this product.

When editing a product there is also a link to edit Group Access Controls, see Section 3.4.4.

Creating New Products

To create a new product:

1. Select “Administration” from the footer and then choose “Products” from the main administration page.
2. Select the “Add” link in the bottom right.
3. Enter the name of the product and a description. The Description field may contain HTML.
4. When the product is created, Bugzilla will give a message stating that a component must be created before any bugs can be entered against the new product. Follow the link to create a new component. See Components for more information.

Editing Products

To edit an existing product, click the "Products" link from the "Administration" page. If the ’useclassification’ parameter is turned on, a table of existing classifications is displayed, including an "Unclassified" category. The table indicates how many products are in each classification. Click on the classification name to see its products. If the ’useclassification’ parameter is not in use, the table lists all products directly. The product table summarizes the information about the product defined when the product was created. Click on the product name to edit these properties, and to access links to other product attributes such as the product’s components, versions, milestones, and group access controls.

Adding or Editing Components, Versions and Target Milestones

To edit existing, or add new, Components, Versions or Target Milestones to a Product, select the "Edit Components", "Edit Versions" or "Edit Milestones" links from the "Edit Product" page. A table of existing Components, Versions or Milestones is displayed. Click on a item name to edit the properties of that item. Below the table is a link to add a new Component, Version or Milestone.

For more information on components, see Components.

For more information on versions, see Section 3.6.

For more information on milestones, see Section 3.7.

Assigning Group Controls to Products

On the “Edit Product” page, there is a link called “Edit Group Access Controls”. The settings on this page control the relationship of the groups to the product being edited.

Group Access Controls are an important aspect of using groups for isolating products and restricting access to bugs filed against those products. For more information on groups, including how to create, edit add users to, and alter permission of, see Section 3.15.

After selecting the "Edit Group Access Controls" link from the "Edit Product" page, a table containing all user-defined groups for this Bugzilla installation is displayed. The system groups that are created when Bugzilla is installed are not applicable to Group Access Controls. Below is description of what each of these fields means.

Groups may be applicable (e.g bugs in this product can be associated with this group) , default (e.g. bugs in this product are in this group by default), and mandatory (e.g. bugs in this product must be associated with this group) for each product. Groups can also control access to bugs for a given product, or be used to make bugs for a product totally read-only unless the group restrictions are met. The best way to understand these relationships is by example. See Section 3.4.4.1 for examples of product and group relationships.

Note
Products and Groups are not limited to a one-to-one relationship. Multiple groups can be associated with the same product, and groups can be associated with more than one product.

If any group has *Entry* selected, then the product will restrict bug entry to only those users who are members of *all* the groups with *Entry* selected.

If any group has *Canedit* selected, then the product will be read-only for any users who are not members of *all* of the groups with *Canedit* selected. *Only* users who are members of all the *Canedit* groups will be able to edit bugs for this product. This is an additional restriction that enables finer-grained control over products rather than just all-or-nothing access levels.

The following settings let you choose privileges on a *per-product basis*. This is a convenient way to give privileges to some users for some products only, without having to give them global privileges which would affect all products.

Any group having *editcomponents* selected allows users who are in this group to edit all aspects of this product, including components, milestones and versions.

Any group having *canconfirm* selected allows users who are in this group to confirm bugs in this product.

Any group having *editbugs* selected allows users who are in this group to edit all fields of bugs in this product.

The *MemberControl* and *OtherControl* are used in tandem to determine which bugs will be placed in this group. The only allowable combinations of these two parameters are listed in a table on the "Edit Group Access Controls" page. Consult this table for details on how these fields can be used. Examples of different uses are described below.

Common Applications of Group Controls

The use of groups is best explained by providing examples that illustrate configurations for common use cases. The examples follow a common syntax: *Group: Entry, MemberControl, OtherControl, CanEdit, EditComponents, CanConfirm, EditBugs*. Where "Group" is the name of the group being edited for this product. The other fields all correspond to the table on the "Edit Group Access Controls" page. If any of these options are not listed, it means they are not checked.

Basic Product/Group Restriction

Suppose there is a product called "Bar". The "Bar" product can only have bugs entered against it by users in the group "Foo". Additionally, bugs filed against product "Bar" must stay restricted to users to "Foo" at all times. Furthermore, only members of group "Foo" can edit bugs filed against product "Bar", even if other users could see the bug. This arrangement would achieved by the following:

```
Product Bar:
foo: ENTRY, MANDATORY/MANDATORY, CANEDIT
```

Perhaps such strict restrictions are not needed for product "Bar". A more lenient way to configure product "Bar" and group "Foo" would be:

```
Product Bar:
foo: ENTRY, SHOWN/SHOWN, EDITCOMPONENTS, CANCONFIRM, EDITBUGS
```

The above indicates that for product "Bar", members of group "Foo" can enter bugs. Any one with permission to edit a bug against product "Bar" can put the bug in group "Foo", even if they themselves are not in "Foo". Anyone in group "Foo" can edit all aspects of the components of product "Bar", can confirm bugs against product "Bar", and can edit all fields of any bug against product "Bar".

General User Access With Security Group

To permit any user to file bugs against "Product A", and to permit any user to submit those bugs into a group called "Security":

```
Product A:
security: SHOWN/SHOWN
```

General User Access With A Security Product

To permit any user to file bugs against product called "Security" while keeping those bugs from becoming visible to anyone outside the group "SecurityWorkers" (unless a member of the "SecurityWorkers" group removes that restriction):

```
Product Security:
securityworkers: DEFAULT/MANDATORY
```

Product Isolation With a Common Group

To permit users of "Product A" to access the bugs for "Product A", users of "Product B" to access the bugs for "Product B", and support staff, who are members of the "Support Group" to access both, three groups are needed:

1. Support Group: Contains members of the support staff.
2. AccessA Group: Contains users of product A and the Support group.
3. AccessB Group: Contains users of product B and the Support group.

Once these three groups are defined, the product group controls can be set to:

```
Product A:
AccessA: ENTRY, MANDATORY/MANDATORY
Product B:
AccessB: ENTRY, MANDATORY/MANDATORY
```

Perhaps the "Support Group" wants more control. For example, the "Support Group" could be permitted to make bugs inaccessible to users of both groups "AccessA" and "AccessB". Then, the "Support Group" could be permitted to publish bugs relevant to all users in a third product (let's call it "Product Common") that is read-only to anyone outside the "Support Group". In this way the "Support Group" could control bugs that should be seen by both groups. That configuration would be:

```
Product A:
AccessA: ENTRY, MANDATORY/MANDATORY
Support: SHOWN/NA
Product B:
AccessB: ENTRY, MANDATORY/MANDATORY
Support: SHOWN/NA
Product Common:
Support: ENTRY, DEFAULT/MANDATORY, CANEDIT
```

Make a Product Read Only

Sometimes a product is retired and should no longer have new bugs filed against it (for example, an older version of a software product that is no longer supported). A product can be made read-only by creating a group called "readonly" and adding products to the group as needed:

```
Product A:
ReadOnly: ENTRY, NA/NA, CANEDIT
```

Note
For more information on Groups outside of how they relate to products see Section 3.15.

Components

Components are subsections of a Product. E.g. the computer game you are designing may have a "UI" component, an "API" component, a "Sound System" component, and a "Plugins" component, each overseen by a different programmer. It often makes sense to divide Components in Bugzilla according to the natural divisions of responsibility within your Product or company.

Each component has a default assignee and (if you turned it on in the parameters), a QA Contact. The default assignee should be the primary person who fixes bugs in that component. The QA Contact should be the person who will ensure these bugs are completely fixed. The Assignee, QA Contact, and Reporter will get email when new bugs are created in this Component and when these bugs change. Default Assignee and Default QA Contact fields only dictate the *default assignments*; these can be changed on bug submission, or at any later point in a bug's life.

To create a new Component:

1. Select the "Edit components" link from the "Edit product" page
2. Select the "Add" link in the bottom right.
3. Fill out the "Component" field, a short "Description", the "Default Assignee", "Default CC List" and "Default QA Contact" (if enabled). The "Component Description" field may contain a limited subset of HTML tags. The "Default Assignee" field must be a login name already existing in the Bugzilla database.

Versions

Versions are the revisions of the product, such as "Flinders 3.1", "Flinders 95", and "Flinders 2000". Version is not a multi-select field; the usual practice is to select the earliest version known to have the bug.

To create and edit Versions:

1. From the "Edit product" screen, select "Edit Versions"
2. You will notice that the product already has the default version "undefined". Click the "Add" link in the bottom right.
3. Enter the name of the Version. This field takes text only. Then click the "Add" button.

Milestones

Milestones are "targets" that you plan to get a bug fixed by. For example, you have a bug that you plan to fix for your 3.0 release, it would be assigned the milestone of 3.0.

Note
Milestone options will only appear for a Product if you turned on the "usetargetmilestone" parameter in the "Bug Fields" tab of the "Parameters" page.

To create new Milestones, and set Default Milestones:

1. Select "Edit milestones" from the "Edit product" page.
2. Select "Add" in the bottom right corner.
3. Enter the name of the Milestone in the "Milestone" field. You can optionally set the "sortkey", which is a positive or negative number (-32768 to 32767) that defines where in the list this particular milestone appears. This is because milestones often do not occur in alphanumeric order For example, "Future" might be after "Release 1.2". Select "Add".

Flags

Flags are a way to attach a specific status to a bug or attachment, either “+” or “-”. The meaning of these symbols depends on the text the flag itself, but contextually they could mean pass/fail, accept/reject, approved/denied, or even a simple yes/no. If your site allows requestable flags, then users may set a flag to “?” as a request to another user that they look at the bug/attachment, and set the flag to its correct status.

A Simple Example

A developer might want to ask their manager, “Should we fix this bug before we release version 2.0?” They might want to do this for a *lot* of bugs, so it would be nice to streamline the process...

In Bugzilla, it would work this way:

1. The Bugzilla administrator creates a flag type called “blocking2.0” that shows up on all bugs in your product.

   It shows up on the “Show Bug” screen as the text “blocking2.0” with a drop-down box next to it. The drop-down box contains four values: an empty space, “?”, “-”, and “+”.
2. The developer sets the flag to “?”.
3. The manager sees the `blocking2.0` flag with a “?” value.
4. If the manager thinks the feature should go into the product before version 2.0 can be released, he sets the flag to “+”. Otherwise, he sets it to “-”.
5. Now, every Bugzilla user who looks at the bug knows whether or not the bug needs to be fixed before release of version 2.0.

About Flags

Values

Flags can have three values:

? A user is requesting that a status be set. (Think of it as ’A question is being asked’.)

− The status has been set negatively. (The question has been answered “no”.)

+ The status has been set positively. (The question has been answered “yes”.)

Actually, there’s a fourth value a flag can have -- “unset” -- which shows up as a blank space. This just means that nobody has expressed an opinion (or asked someone else to express an opinion) about this bug or attachment.

Using flag requests

If a flag has been defined as 'requestable', and a user has enough privileges to request it (see below), the user can set the flag's status to "?". This status indicates that someone (a.k.a. "the requester") is asking someone else to set the flag to either "+" or "-".

If a flag has been defined as 'specifically requestable', a text box will appear next to the flag into which the requester may enter a Bugzilla username. That named person (a.k.a. "the requestee") will receive an email notifying them of the request, and pointing them to the bug/attachment in question.

If a flag has *not* been defined as 'specifically requestable', then no such text-box will appear. A request to set this flag cannot be made of any specific individual, but must be asked "to the wind". A requester may "ask the wind" on any flag simply by leaving the text-box blank.

Two Types of Flags

Flags can go in two places: on an attachment, or on a bug.

Attachment Flags

Attachment flags are used to ask a question about a specific attachment on a bug.

Many Bugzilla installations use this to request that one developer "review" another developer's code before they check it in. They attach the code to a bug report, and then set a flag on that attachment called "review" to `review?boss@domain.com`. boss@domain.com is then notified by email that he has to check out that attachment and approve it or deny it.

For a Bugzilla user, attachment flags show up in three places:

1. On the list of attachments in the "Show Bug" screen, you can see the current state of any flags that have been set to ?, +, or -. You can see who asked about the flag (the requester), and who is being asked (the requestee).
2. When you "Edit" an attachment, you can see any settable flag, along with any flags that have already been set. This "Edit Attachment" screen is where you set flags to ?, -, +, or unset them.
3. Requests are listed in the "Request Queue", which is accessible from the "My Requests" link (if you are logged in) or "Requests" link (if you are logged out) visible in the footer of all pages.

Bug Flags

Bug flags are used to set a status on the bug itself. You can see Bug Flags in the "Show Bug" and "Requests" screens, as described above.

Only users with enough privileges (see below) may set flags on bugs. This doesn't necessarily include the assignee, reporter, or users with the `editbugs` permission.

Administering Flags

If you have the "editcomponents" permission, you can edit Flag Types from the main administration page. Clicking the "Flags" link will bring you to the "Administer Flag Types" page. Here, you can select whether you want to create (or edit) a Bug flag, or an Attachment flag.

No matter which you choose, the interface is the same, so we'll just go over it once.

Editing a Flag

To edit a flag's properties, just click the flag's name. That will take you to the same form as described below (Section 3.8.5.2).

Creating a Flag

When you click on the "Create a Flag Type for..." link, you will be presented with a form. Here is what the fields in the form mean:

Name

This is the name of the flag. This will be displayed to Bugzilla users who are looking at or setting the flag. The name may contain any valid Unicode characters except commas and spaces.

Description

The description describes the flag in more detail. It is visible in a tooltip when hovering over a flag either in the "Show Bug" or "Edit Attachment" pages. This field can be as long as you like, and can contain any character you want.

Category

Default behaviour for a newly-created flag is to appear on products and all components, which is why "__Any__:__Any__" is already entered in the "Inclusions" box. If this is not your desired behaviour, you must either set some exclusions (for products on which you don't want the flag to appear), or you must remove "__Any__:__Any__" from the Inclusions box and define products/components specifically for this flag.

To create an Inclusion, select a Product from the top drop-down box. You may also select a specific component from the bottom drop-down box. (Setting "__Any__" for Product translates to, "all the products in this Bugzilla". Selecting "__Any__" in the Component field means "all components in the selected product.") Selections made, press "Include", and your Product/Component pairing will show up in the "Inclusions" box on the right.

To create an Exclusion, the process is the same; select a Product from the top drop-down box, select a specific component if you want one, and press "Exclude". The Product/Component pairing will show up in the "Exclusions" box on the right.

This flag *will* and *can* be set for any products/components that appearing in the "Inclusions" box (or which fall under the appropriate "__Any__"). This flag *will not* appear (and therefore cannot be set) on any products appearing in the "Exclusions" box. *IMPORTANT: Exclusions override inclusions.*

You may select a Product without selecting a specific Component, but you can't select a Component without a Product, or to select a Component that does not belong to the named Product. If you do so, Bugzilla will display an error message, even if all your products have a component by that name.

Example: Let's say you have a product called "Jet Plane" that has thousands of components. You want to be able to ask if a problem should be fixed in the next model of plane you release. We'll call the flag "fixInNext". But, there's one component in "Jet Plane," called "Pilot." It doesn't make sense to release a new pilot, so you don't want to have the flag show up in that component. So, you include "Jet Plane:__Any__" and you exclude "Jet Plane:Pilot".

Sort Key

Flags normally show up in alphabetical order. If you want them to show up in a different order, you can use this key set the order on each flag. Flags with a lower sort key will appear before flags with a higher sort key. Flags that have the same sort key will be sorted alphabetically, but they will still be after flags with a lower sort key, and before flags with a higher sort key.

Example: I have AFlag (Sort Key 100), BFlag (Sort Key 10), CFlag (Sort Key 10), and DFlag (Sort Key 1). These show up in the order: DFlag, BFlag, CFlag, AFlag.

Active

Sometimes, you might want to keep old flag information in the Bugzilla database, but stop users from setting any new flags of this type. To do this, uncheck "active". Deactivated flags will still show up in the UI if they are ?, +, or -, but they may only be cleared (unset), and cannot be changed to a new value. Once a deactivated flag is cleared, it will completely disappear from a bug/attachment, and cannot be set again.

Requestable

New flags are, by default, “requestable”, meaning that they offer users the “?” option, as well as “+” and “-”. To remove the ? option, uncheck “requestable”.

Specifically Requestable

By default this box is checked for new flags, meaning that users may make flag requests of specific individuals. Unchecking this box will remove the text box next to a flag; if it is still requestable, then requests may only be made “to the wind.” Removing this after specific requests have been made will not remove those requests; that data will stay in the database (though it will no longer appear to the user).

Multiplicable

Any flag with “Multiplicable” set (default for new flags is ’on’) may be set more than once. After being set once, an unset flag of the same type will appear below it with “addl.” (short for “additional”) before the name. There is no limit to the number of times a Multiplicable flags may be set on the same bug/attachment.

CC List

If you want certain users to be notified every time this flag is set to ?, -, +, or unset, add them here. This is a comma-separated list of email addresses that need not be restricted to Bugzilla usernames.

Grant Group

When this field is set to some given group, only users in the group can set the flag to “+” and “-”. This field does not affect who can request or cancel the flag. For that, see the “Request Group” field below. If this field is left blank, all users can set or delete this flag. This field is useful for restricting which users can approve or reject requests.

Request Group

When this field is set to some given group, only users in the group can request or cancel this flag. Note that this field has no effect if the “grant group” field is empty. You can set the value of this field to a different group, but both fields have to be set to a group for this field to have an effect.

Deleting a Flag

When you are at the “Administer Flag Types” screen, you will be presented with a list of Bug flags and a list of Attachment Flags.

To delete a flag, click on the “Delete” link next to the flag description.

Warning
Once you delete a flag, it is *gone* from your Bugzilla. All the data for that flag will be deleted. Everywhere that flag was set, it will disappear, and you cannot get that data back. If you want to keep flag data, but don’t want anybody to set any new flags or change current flags, unset “active” in the flag Edit form.

Keywords

The administrator can define keywords which can be used to tag and categorise bugs. For example, the keyword "regression" is commonly used. A company might have a policy stating all regressions must be fixed by the next release - this keyword can make tracking those bugs much easier.

Keywords are global, rather than per-product. If the administrator changes a keyword currently applied to any bugs, the keyword cache must be rebuilt using the Section 3.16 script. Currently keywords cannot be marked obsolete to prevent future usage.

Keywords can be created, edited or deleted by clicking the "Keywords" link in the admin page. There are two fields for each keyword - the keyword itself and a brief description. Once created, keywords can be selected and applied to individual bugs in that bug's "Details" section.

Custom Fields

The release of Bugzilla 3.0 added the ability to create Custom Fields. Custom Fields are treated like any other field - they can be set in bugs and used for search queries. Administrators should keep in mind that adding too many fields can make the user interface more complicated and harder to use. Custom Fields should be added only when necessary and with careful consideration.

Tip
Before adding a Custom Field, make sure that Bugzilla cannot already do the desired behavior. Many Bugzilla options are not enabled by default, and many times Administrators find that simply enabling certain options that already exist is sufficient.

Administrators can manage Custom Fields using the "Custom Fields" link on the Administration page. The Custom Fields administration page displays a list of Custom Fields, if any exist, and a link to "Add a new custom field".

Adding Custom Fields

To add a new Custom Field, click the "Add a new custom field" link. This page displays several options for the new field, described below.

The following attributes must be set for each new custom field:

- *Name:* The name of the field in the database, used internally. This name MUST begin with "cf_" to prevent confusion with standard fields. If this string is omitted, it will be automatically added to the name entered.
- *Description:* A brief string which is used as the label for this Custom Field. That is the string that users will see, and should be short and explicit.
- *Type:* The type of field to create. There are several types available:

  Bug ID: A field where you can enter the ID of another bug from the same Bugzilla installation. To point to a bug in a remote installation, use the See Also field instead.

  Large Text Box: A multiple line box for entering free text.

  Free Text: A single line box for entering free text.

  Multiple-Selection Box: A list box where multiple options can be selected. After creating this field, it must be edited to add the selection options. See Section 3.11.1 for information about editing legal values.

  Drop Down: A list box where only one option can be selected. After creating this field, it must be edited to add the selection options. See Section 3.11.1 for information about editing legal values.

  Date/Time: A date field. This field appears with a calendar widget for choosing the date.

- *Sortkey:* Integer that determines in which order Custom Fields are displayed in the User Interface, especially when viewing a bug. Fields with lower values are displayed first.

- *Reverse Relationship Description:* When the custom field is of type “Bug ID”, you can enter text here which will be used as label in the referenced bug to list bugs which point to it. This gives you the ability to have a mutual relationship between two bugs.
- *Can be set on bug creation:* Boolean that determines whether this field can be set on bug creation. If not selected, then a bug must be created before this field can be set. See Section 5.6 for information about filing bugs.
- *Displayed in bugmail for new bugs:* Boolean that determines whether the value set on this field should appear in bugmail when the bug is filed. This attribute has no effect if the field cannot be set on bug creation.
- *Is obsolete:* Boolean that determines whether this field should be displayed at all. Obsolete Custom Fields are hidden.
- *Is mandatory:* Boolean that determines whether this field must be set. For single and multi-select fields, this means that a (non-default) value must be selected, and for text and date fields, some text must be entered.
- *Field only appears when:* A custom field can be made visible when some criteria is met. For instance, when the bug belongs to one or more products, or when the bug is of some given severity. If left empty, then the custom field will always be visible, in all bugs.
- *Field that controls the values that appear in this field:* When the custom field is of type “Drop Down” or “Multiple-Selection Box”, you can restrict the availability of the values of the custom field based on the value of another field. This criteria is independent of the criteria used in the “Field only appears when” setting. For instance, you may decide that some given value “valueY” is only available when the bug status is RESOLVED while the value “valueX” should always be listed. Once you have selected the field which should control the availability of the values of this custom field, you can edit values of this custom field to set the criteria, see Section 3.11.1.

Editing Custom Fields

As soon as a Custom Field is created, its name and type cannot be changed. If this field is a drop down menu, its legal values can be set as described in Section 3.11.1. All other attributes can be edited as described above.

Deleting Custom Fields

Only custom fields which are marked as obsolete, and which never have been used, can be deleted completely (else the integrity of the bug history would be compromised). For custom fields marked as obsolete, a "Delete" link will appear in the “Action” column. If the custom field has been used in the past, the deletion will be rejected. But marking the field as obsolete is sufficient to hide it from the user interface entirely.

Legal Values

Legal values for the operating system, platform, bug priority and severity, custom fields of type “Drop Down” and “Multiple-Selection Box” (see Section 3.10), as well as the list of valid bug statuses and resolutions can be customized from the same interface. You can add, edit, disable and remove values which can be used with these fields.

Viewing/Editing legal values

Editing legal values requires “admin” privileges. Select "Field Values" from the Administration page. A list of all fields, both system fields and Custom Fields, for which legal values can be edited appears. Click a field name to edit its legal values.

There is no limit to how many values a field can have, but each value must be unique to that field. The sortkey is important to display these values in the desired order.

When the availability of the values of a custom field is controlled by another field, you can select from here which value of the other field must be set for the value of the custom field to appear.

Deleting legal values

Legal values from Custom Fields can be deleted, but only if the following two conditions are respected:

1. The value is not used by default for the field.
2. No bug is currently using this value.

If any of these conditions is not respected, the value cannot be deleted. The only way to delete these values is to reassign bugs to another value and to set another value as default for the field.

Bug Status Workflow

The bug status workflow is no longer hardcoded but can be freely customized from the web interface. Only one bug status cannot be renamed nor deleted, UNCONFIRMED, but the workflow involving it is free. The configuration page displays all existing bug statuses twice, first on the left for bug statuses we come from and on the top for bug statuses we move to. If the checkbox is checked, then the transition between the two bug statuses is legal, else it's forbidden independently of your privileges. The bug status used for the "duplicate_or_move_bug_status" parameter must be part of the workflow as that is the bug status which will be used when duplicating or moving a bug, so it must be available from each bug status.

When the workflow is set, the "View Current Triggers" link below the table lets you set which transitions require a comment from the user.

Voting

All of the code for voting in Bugzilla has been moved into an extension, called "Voting", in the `extensions/Voting/` directory. To enable it, you must remove the `disabled` file from that directory, and run `checksetup.pl`.

Voting allows users to be given a pot of votes which they can allocate to bugs, to indicate that they'd like them fixed. This allows developers to gauge user need for a particular enhancement or bugfix. By allowing bugs with a certain number of votes to automatically move from "UNCONFIRMED" to "CONFIRMED", users of the bug system can help high-priority bugs garner attention so they don't sit for a long time awaiting triage.

To modify Voting settings:

1. Navigate to the "Edit product" screen for the Product you wish to modify
2. *Maximum Votes per person*: Setting this field to "0" disables voting.
3. *Maximum Votes a person can put on a single bug*: It should probably be some number lower than the "Maximum votes per person". Don't set this field to "0" if "Maximum votes per person" is non-zero; that doesn't make any sense.
4. *Number of votes a bug in this product needs to automatically get out of the UNCONFIRMED state*: Setting this field to "0" disables the automatic move of bugs from UNCONFIRMED to CONFIRMED.
5. Once you have adjusted the values to your preference, click "Update".

Quips

Quips are small text messages that can be configured to appear next to search results. A Bugzilla installation can have its own specific quips. Whenever a quip needs to be displayed, a random selection is made from the pool of already existing quips.

Quip submission is controlled by the *quip_list_entry_control* parameter. It has several possible values: open, moderated, or closed. In order to enable quips approval you need to set this parameter to "moderated". In this way, users are free to submit quips for addition but an administrator must explicitly approve them before they are actually used.

In order to see the user interface for the quips, it is enough to click on a quip when it is displayed together with the search results. Or it can be seen directly in the browser by visiting the quips.cgi URL (prefixed with the usual web location of the Bugzilla installation). Once the quip interface is displayed, it is enough to click the "view and edit the whole quip list" in order to see the administration page. A page with all the quips available in the database will be displayed.

Next to each quip there is a checkbox, under the "Approved" column. Quips who have this checkbox checked are already approved and will appear next to the search results. The ones that have it unchecked are still preserved in the database but they will not appear on search results pages. User submitted quips have initially the checkbox unchecked.

Also, there is a delete link next to each quip, which can be used in order to permanently delete a quip.

Display of quips is controlled by the *display_quips* user preference. Possible values are "on" and "off".

Groups and Group Security

Groups allow for separating bugs into logical divisions. Groups are typically used to isolate bugs that should only be seen by certain people. For example, a company might create a different group for each one of its customers or partners. Group permissions could be set so that each partner or customer would only have access to their own bugs. Or, groups might be used to create variable access controls for different departments within an organization. Another common use of groups is to associate groups with products, creating isolation and access control on a per-product basis.

Groups and group behaviors are controlled in several places:

1. The group configuration page. To view or edit existing groups, or to create new groups, access the "Groups" link from the "Administration" page. This section of the manual deals primarily with the aspect of group controls accessed on this page.
2. Global configuration parameters. Bugzilla has several parameters that control the overall default group behavior and restriction levels. For more information on the parameters that control group behavior globally, see Section 3.1.9.
3. Product association with groups. Most of the functionality of groups and group security is controlled at the product level. Some aspects of group access controls for products are discussed in this section, but for more detail see Section 3.4.4.
4. Group access for users. See Section 3.15.3 for details on how users are assigned group access.

Group permissions are such that if a bug belongs to a group, only members of that group can see the bug. If a bug is in more than one group, only members of *all* the groups that the bug is in can see the bug. For information on granting read-only access to certain people and full edit access to others, see Section 3.4.4.

Note

By default, bugs can also be seen by the Assignee, the Reporter, and by everyone on the CC List, regardless of whether or not the bug would typically be viewable by them. Visibility to the Reporter and CC List can be overridden (on a per-bug basis) by bringing up the bug, finding the section that starts with “Users in the roles selected below...” and un-checking the box next to either ’Reporter’ or ’CC List’ (or both).

Creating Groups

To create a new group, follow the steps below:

1. Select the “Administration” link in the page footer, and then select the “Groups” link from the Administration page.
2. A table of all the existing groups is displayed. Below the table is a description of all the fields. To create a new group, select the “Add Group” link under the table of existing groups.
3. There are five fields to fill out. These fields are documented below the form. Choose a name and description for the group. Decide whether this group should be used for bugs (in all likelihood this should be selected). Optionally, choose a regular expression that will automatically add any matching users to the group, and choose an icon that will help identify user comments for the group. The regular expression can be useful, for example, to automatically put all users from the same company into one group (if the group is for a specific customer or partner).

Note
If “User RegExp” is filled out, users whose email addresses match the regular expression will automatically be members of the group as long as their email addresses continue to match the regular expression. If their email address changes and no longer matches the regular expression, they will be removed from the group. Versions 2.16 and older of Bugzilla did not automatically remove users who’s email addresses no longer matched the RegExp.

Warning
If specifying a domain in the regular expression, end the regexp with a "$". Otherwise, when granting access to "@mycompany\.com", access will also be granted to ’badperson@mycompany.com.cracker.net’. Use the syntax, ’@mycompany\.com$’ for the regular expression.

4. After the new group is created, it can be edited for additional options. The "Edit Group" page allows for specifying other groups that should be included in this group and which groups should be permitted to add and delete users from this group. For more details, see Section 3.15.2.

Editing Groups and Assigning Group Permissions

To access the "Edit Groups" page, select the “Administration” link in the page footer, and then select the “Groups” link from the Administration page. A table of all the existing groups is displayed. Click on a group name you wish to edit or control permissions for.

The "Edit Groups" page contains the same five fields present when creating a new group. Below that are two additional sections, "Group Permissions," and "Mass Remove". The "Mass Remove" option simply removes all users from the group who match the regular expression entered. The "Group Permissions" section requires further explanation.

The "Group Permissions" section on the "Edit Groups" page contains four sets of permissions that control the relationship of this group to other groups. If the ’usevisibilitygroups’ parameter is in use (see Section 3.1) two additional sets of permissions are displayed. Each set consists of two select boxes. On the left, a select box with a list of all existing groups. On the right, a select box listing all groups currently selected for this permission setting (this box will be empty for new groups). The way these controls allow groups to relate to one another is called *inheritance*. Each of the six permissions is described below.

Groups That Are a Member of This Group Members of any groups selected here will automatically have membership in this group. In other words, members of any selected group will inherit membership in this group.

Groups That This Group Is a Member Of Members of this group will inherit membership to any group selected here. For example, suppose the group being edited is an Admin group. If there are two products (Product1 and Product2) and each product has its own group (Group1 and Group2), and the Admin group should have access to both products, simply select both Group1 and Group2 here.

Groups That Can Grant Membership in This Group The members of any group selected here will be able add users to this group, even if they themselves are not in this group.

Groups That This Group Can Grant Membership In Members of this group can add users to any group selected here, even if they themselves are not in the selected groups.

Groups That Can See This Group Members of any selected group can see the users in this group. This setting is only visible if the ’usevisibilitygroups’ parameter is enabled on the Bugzilla Configuration page. See Section 3.1 for information on configuring Bugzilla.

Groups That This Group Can See Members of this group can see members in any of the selected groups. This setting is only visible if the ’usevisibilitygroups’ parameter is enabled on the the Bugzilla Configuration page. See Section 3.1 for information on configuring Bugzilla.

Assigning Users to Groups

A User can become a member of a group in several ways:

1. The user can be explicitly placed in the group by editing the user's profile. This can be done by accessing the "Users" page from the "Administration" page. Use the search form to find the user you want to edit group membership for, and click on their email address in the search results to edit their profile. The profile page lists all the groups, and indicates if the user is a member of the group either directly or indirectly. More information on indirect group membership is below. For more details on User administration, see Section 3.2.
2. The group can include another group of which the user is a member. This is indicated by square brackets around the checkbox next to the group name in the user's profile. See Section 3.15.2 for details on group inheritance.
3. The user's email address can match the regular expression that has been specified to automatically grant membership to the group. This is indicated by "*" around the check box by the group name in the user's profile. See Section 3.15.1 for details on the regular expression option when creating groups.

Assigning Group Controls to Products

The primary functionality of groups is derived from the relationship of groups to products. The concepts around segregating access to bugs with product group controls can be confusing. For details and examples on this topic, see Section 3.4.4.

Checking and Maintaining Database Integrity

Over time it is possible for the Bugzilla database to become corrupt or to have anomalies. This could happen through normal usage of Bugzilla, manual database administration outside of the Bugzilla user interface, or from some other unexpected event. Bugzilla includes a "Sanity Check" script that can perform several basic database checks, and repair certain problems or inconsistencies.

To run the "Sanity Check" script, log in as an Administrator and click the "Sanity Check" link in the admin page. Any problems that are found will be displayed in red letters. If the script is capable of fixing a problem, it will present a link to initiate the fix. If the script cannot fix the problem it will require manual database administration or recovery.

The "Sanity Check" script can also be run from the command line via the perl script `sanitycheck.pl`. The script can also be run as a **cron** job. Results will be delivered by email.

The "Sanity Check" script should be run on a regular basis as a matter of best practice.

Warning
The "Sanity Check" script is no substitute for a competent database administrator. It is only designed to check and repair basic database problems.

Chapter 4

Bugzilla Security

While some of the items in this chapter are related to the operating system Bugzilla is running on or some of the support software required to run Bugzilla, it is all related to protecting your data. This is not intended to be a comprehensive guide to securing Linux, Apache, MySQL, or any other piece of software mentioned. There is no substitute for active administration and monitoring of a machine. The key to good security is actually right in the middle of the word: *U R It*.

While programmers in general always strive to write secure code, accidents can and do happen. The best approach to security is to always assume that the program you are working with isn't 100% secure and restrict its access to other parts of your machine as much as possible.

Operating System

TCP/IP Ports

The TCP/IP standard defines more than 65,000 ports for sending and receiving traffic. Of those, Bugzilla needs exactly one to operate (different configurations and options may require up to 3). You should audit your server and make sure that you aren't listening on any ports you don't need to be. It's also highly recommended that the server Bugzilla resides on, along with any other machines you administer, be placed behind some kind of firewall.

System User Accounts

Many *daemons*, such as Apache's `httpd` or MySQL's `mysqld`, run as either "root" or "nobody". This is even worse on Windows machines where the majority of *services* run as "SYSTEM". While running as "root" or "SYSTEM" introduces obvious security concerns, the problems introduced by running everything as "nobody" may not be so obvious. Basically, if you run every daemon as "nobody" and one of them gets compromised it can compromise every other daemon running as "nobody" on your machine. For this reason, it is recommended that you create a user account for each daemon.

Note
You will need to set the `webservergroup` option in `localconfig` to the group your web server runs as. This will allow `./checksetup.pl` to set file permissions on Unix systems so that nothing is world-writable.

The `chroot` Jail

If your system supports it, you may wish to consider running Bugzilla inside of a `chroot` jail. This option provides unprecedented security by restricting anything running inside the jail from accessing any information outside of it. If you wish to use this option, please consult the documentation that came with your system.

Web server

Disabling Remote Access to Bugzilla Configuration Files

There are many files that are placed in the Bugzilla directory area that should not be accessible from the web server. Because of the way Bugzilla is currently layed out, the list of what should and should not be accessible is rather complicated. A quick way is to run `testserver.pl` to check if your web server serves Bugzilla files as expected. If not, you may want to follow the few steps below.

Tip
Bugzilla ships with the ability to create *`.htaccess`* files that enforce these rules. Instructions for enabling these directives in Apache can be found in Section 2.2.4.1

- In the main Bugzilla directory, you should:
  - Block: `*.pl`, `*localconfig*`
- In `data`:
  - Block everything
- In `data/webdot`:
  - If you use a remote webdot server:
    - Block everything
    - But allow `*.dot` only for the remote webdot server
  - Otherwise, if you use a local GraphViz:
    - Block everything
    - But allow: `*.png`, `*.gif`, `*.jpg`, `*.map`
  - And if you don't use any dot:
    - Block everything
- In `Bugzilla`:
  - Block everything
- In `template`:
  - Block everything

Be sure to test that data that should not be accessed remotely is properly blocked. Of particular interest is the localconfig file which contains your database password. Also, be aware that many editors create temporary and backup files in the working directory and that those should also not be accessible. For more information, see bug 186383[1] or Bugtraq ID 6501[2]. To test, simply run `testserver.pl`, as said above.

Tip
Be sure to check Section 2.2.4 for instructions specific to the web server you use.

[1] `http://bugzilla.mozilla.org/show_bug.cgi?id=186383`
[2] `http://online.securityfocus.com/bid/6501`

Bugzilla

Prevent users injecting malicious Javascript

If you installed Bugzilla version 2.22 or later from scratch, then the *utf8* parameter is switched on by default. This makes Bugzilla explicitly set the character encoding, following a CERT advisory[3] recommending exactly this. The following therefore does not apply to you; just keep *utf8* turned on.

If you've upgraded from an older version, then it may be possible for a Bugzilla user to take advantage of character set encoding ambiguities to inject HTML into Bugzilla comments. This could include malicious scripts. This is because due to internationalization concerns, we are unable to turn the *utf8* parameter on by default for upgraded installations. Turning it on manually will prevent this problem.

[3] http://www.cert.org/tech_tips/malicious_code_mitigation.html#3

Chapter 5

Using Bugzilla

Introduction

This section contains information for end-users of Bugzilla. There is a Bugzilla test installation, called Landfill[1], which you are welcome to play with (if it's up). However, not all of the Bugzilla installations there will necessarily have all Bugzilla features enabled, and different installations run different versions, so some things may not quite work as this document describes.

Frequently Asked Questions (FAQ) are available and answered on wiki.mozilla.org[2]. They may cover some questions you have which are left unanswered.

Create a Bugzilla Account

If you want to use Bugzilla, first you need to create an account. Consult with the administrator responsible for your installation of Bugzilla for the URL you should use to access it. If you're test-driving Bugzilla, use this URL: `http://landfill.bugzilla.org/bugzilla-4.4-branch/`.

1. On the home page `index.cgi`, click the "Open a new Bugzilla account" link, or the "New Account" link available in the footer of pages. Now enter your email address, then click the "Send" button.

   Note
   If none of these links is available, this means that the administrator of the installation has disabled self-registration. This means that only an administrator can create accounts for other users. One reason could be that this installation is private.

   Note
   Also, if only some users are allowed to create an account on the installation, you may see these links but your registration may fail if your email address doesn't match the ones accepted by the installation. This is another way to restrict who can access and edit bugs in this installation.

2. Within moments, and if your registration is accepted, you should receive an email to the address you provided, which contains your login name (generally the same as the email address), and two URLs with a token (a random string generated by the installation) to confirm, respectively cancel, your registration. This is a way to prevent users from abusing the generation of user accounts, for instance by entering inexistent email addresses, or email addresses which do not belong to them.

[1] `http://landfill.bugzilla.org/`
[2] `http://wiki.mozilla.org/Bugzilla:FAQ`

3. By default, you have 3 days to confirm your registration. Past this timeframe, the token is invalidated and the registration is automatically canceled. You can also cancel this registration sooner by using the appropriate URL in the email you got.
4. If you confirm your registration, Bugzilla will ask you your real name (optional, but recommended) and your password, which must be between 3 and 16 characters long.
5. Now all you need to do is to click the “Log In” link in the footer at the bottom of the page in your browser, enter your email address and password you just chose into the login form, and click the “Log in” button.

You are now logged in. Bugzilla uses cookies to remember you are logged in so, unless you have cookies disabled or your IP address changes, you should not have to log in again during your session.

Anatomy of a Bug

The core of Bugzilla is the screen which displays a particular bug. It’s a good place to explain some Bugzilla concepts. Bug 1 on Landfill[3] is a good example. Note that the labels for most fields are hyperlinks; clicking them will take you to context-sensitive help on that particular field. Fields marked * may not be present on every installation of Bugzilla.

1. *Product and Component*: Bugs are divided up by Product and Component, with a Product having one or more Components in it. For example, bugzilla.mozilla.org’s "Bugzilla" Product is composed of several Components:

   Administration: Administration of a Bugzilla installation.

   Bugzilla-General: Anything that doesn’t fit in the other components, or spans multiple components.

   Creating/Changing Bugs: Creating, changing, and viewing bugs.

   Documentation: The Bugzilla documentation, including The Bugzilla Guide.

   Email: Anything to do with email sent by Bugzilla.

   Installation: The installation process of Bugzilla.

   Query/Buglist: Anything to do with searching for bugs and viewing the buglists.

   Reporting/Charting: Getting reports from Bugzilla.

   User Accounts: Anything about managing a user account from the user’s perspective. Saved queries, creating accounts, changing passwords, logging in, etc.

   User Interface: General issues having to do with the user interface cosmetics (not functionality) including cosmetic issues, HTML templates, etc.

2. *Status and Resolution:* These define exactly what state the bug is in - from not even being confirmed as a bug, through to being fixed and the fix confirmed by Quality Assurance. The different possible values for Status and Resolution on your installation should be documented in the context-sensitive help for those items.
3. *Assigned To:* The person responsible for fixing the bug.
4. **QA Contact:* The person responsible for quality assurance on this bug.
5. **URL:* A URL associated with the bug, if any.
6. *Summary:* A one-sentence summary of the problem.
7. **Status Whiteboard:* (a.k.a. Whiteboard) A free-form text area for adding short notes and tags to a bug.
8. **Keywords:* The administrator can define keywords which you can use to tag and categorise bugs - e.g. The Mozilla Project has keywords like crash and regression.
9. *Platform and OS:* These indicate the computing environment where the bug was found.
10. *Version:* The "Version" field is usually used for versions of a product which have been released, and is set to indicate which versions of a Component have the particular problem the bug report is about.

[3] http://landfill.bugzilla.org/bugzilla-4.4-branch/show_bug.cgi?id=1

11. *Priority:* The bug assignee uses this field to prioritize his or her bugs. It's a good idea not to change this on other people's bugs.
12. *Severity:* This indicates how severe the problem is - from blocker ("application unusable") to trivial ("minor cosmetic issue"). You can also use this field to indicate whether a bug is an enhancement request.
13. **Target:* (a.k.a. Target Milestone) A future version by which the bug is to be fixed. e.g. The Bugzilla Project's milestones for future Bugzilla versions are 2.18, 2.20, 3.0, etc. Milestones are not restricted to numbers, thought - you can use any text strings, such as dates.
14. *Reporter:* The person who filed the bug.
15. *CC list:* A list of people who get mail when the bug changes.
16. **Time Tracking:* This form can be used for time tracking. To use this feature, you have to be blessed group membership specified by the "timetrackinggroup" parameter.

    Orig. Est.: This field shows the original estimated time.

    Current Est.: This field shows the current estimated time. This number is calculated from "Hours Worked" and "Hours Left".

    Hours Worked: This field shows the number of hours worked.

    Hours Left: This field shows the "Current Est." - "Hours Worked". This value + "Hours Worked" will become the new Current Est.

    %Complete: This field shows what percentage of the task is complete.

    Gain: This field shows the number of hours that the bug is ahead of the "Orig. Est.".

    Deadline: This field shows the deadline for this bug.
17. *Attachments:* You can attach files (e.g. testcases or patches) to bugs. If there are any attachments, they are listed in this section.
18. **Dependencies:* If this bug cannot be fixed unless other bugs are fixed (depends on), or this bug stops other bugs being fixed (blocks), their numbers are recorded here.
19. **Votes:* Whether this bug has any votes.
20. *Additional Comments:* You can add your two cents to the bug discussion here, if you have something worthwhile to say.

Life Cycle of a Bug

The life cycle of a bug, also known as workflow, is customizable to match the needs of your organization, see Section 3.12. Figure 5.1 contains a graphical representation of the default workflow using the default bug statuses. If you wish to customize this image for your site, the diagram file[4] is available in Dia's[5] native XML format.

[4] ../images/bzLifecycle.xml
[5] http://www.gnome.org/projects/dia

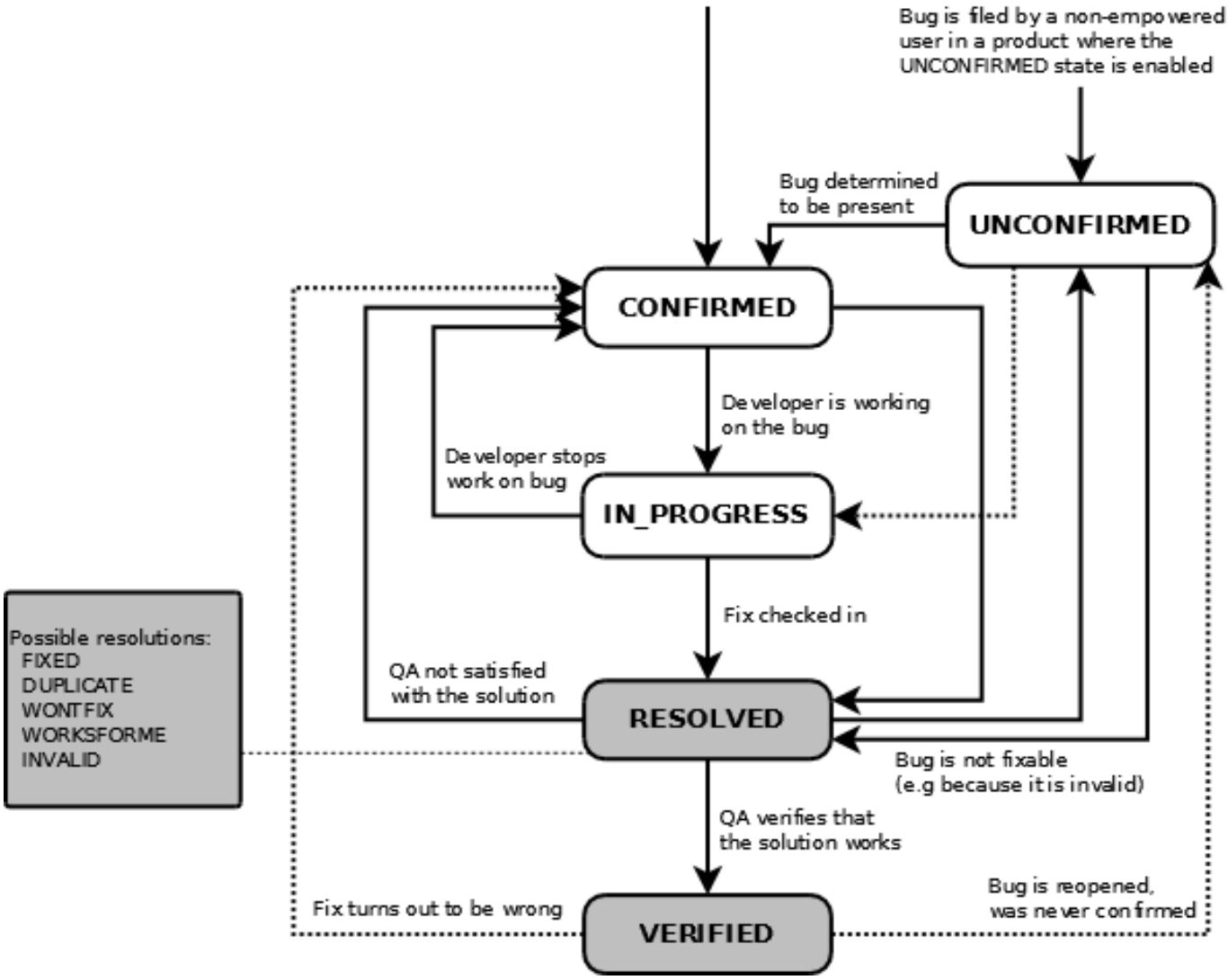

Figure 5.1: Lifecycle of a Bugzilla Bug

Searching for Bugs

The Bugzilla Search page is the interface where you can find any bug report, comment, or patch currently in the Bugzilla system. You can play with it here: `http://landfill.bugzilla.org/bugzilla-4.4-branch/query.cgi`.

The Search page has controls for selecting different possible values for all of the fields in a bug, as described above. For some fields, multiple values can be selected. In those cases, Bugzilla returns bugs where the content of the field matches any one of the selected values. If none is selected, then the field can take any value.

After a search is run, you can save it as a Saved Search, which will appear in the page footer. If you are in the group defined by the "querysharegroup" parameter, you may share your queries with other users, see Saved Searches for more details.

Boolean Charts

Highly advanced querying is done using Boolean Charts.

The boolean charts further restrict the set of results returned by a query. It is possible to search for bugs based on elaborate combinations of criteria.

The simplest boolean searches have only one term. These searches permit the selected left *field* to be compared using a selectable *operator* to a specified *value.* Using the "And," "Or," and "Add Another Boolean Chart" buttons, additional terms can be included in the query, further altering the list of bugs returned by the query.

There are three fields in each row of a boolean search.

- *Field:* the items being searched
- *Operator:* the comparison operator
- *Value:* the value to which the field is being compared

Pronoun Substitution

Sometimes, a query needs to compare a user-related field (such as ReportedBy) with a role-specific user (such as the user running the query or the user to whom each bug is assigned). When the operator is either "is equal to" or "is not equal to", the value can be "%reporter%", "%assignee%", "%qacontact%", or "%user%". The user pronoun refers to the user who is executing the query or, in the case of whining reports, the user who will be the recipient of the report. The reporter, assignee, and qacontact pronouns refer to the corresponding fields in the bug.

Boolean charts also let you type a group name in any user-related field if the operator is either "is equal to", "is not equal to" or "contains the string (exact case)". This will let you query for any member belonging (or not) to the specified group. The group name must be entered following the "%group.foo%" syntax, where "foo" is the group name. So if you are looking for bugs reported by any user being in the "editbugs" group, then you can type "%group.editbugs%".

Negation

At first glance, negation seems redundant. Rather than searching for

> NOT("summary" "contains the string" "foo"),

one could search for

> ("summary" "does not contain the string" "foo").

However, the search

> ("CC" "does not contain the string" "@mozilla.org")

would find every bug where anyone on the CC list did not contain "@mozilla.org" while

> NOT("CC" "contains the string" "@mozilla.org")

would find every bug where there was nobody on the CC list who did contain the string. Similarly, the use of negation also permits complex expressions to be built using terms OR'd together and then negated. Negation permits queries such as

> NOT(("product" "is equal to" "update") OR ("component" "is equal to" "Documentation"))

to find bugs that are neither in the update product or in the documentation component or

> NOT(("commenter" "is equal to" "%assignee%") OR ("component" "is equal to" "Documentation"))

to find non-documentation bugs on which the assignee has never commented.

Multiple Charts

The terms within a single row of a boolean chart are all constraints on a single piece of data. If you are looking for a bug that has two different people cc'd on it, then you need to use two boolean charts. A search for

> ("cc" "contains the string" "foo@") AND ("cc" "contains the string" "@mozilla.org")

would return only bugs with "foo@mozilla.org" on the cc list. If you wanted bugs where there is someone on the cc list containing "foo@" and someone else containing "@mozilla.org", then you would need two boolean charts.

> First chart: ("cc" "contains the string" "foo@")
>
> Second chart: ("cc" "contains the string" "@mozilla.org")

The bugs listed will be only the bugs where ALL the charts are true.

Quicksearch

Quicksearch is a single-text-box query tool which uses metacharacters to indicate what is to be searched. For example, typing "`foo|bar`" into Quicksearch would search for "foo" or "bar" in the summary and status whiteboard of a bug; adding "`: BazProduct`" would search only in that product. You can use it to find a bug by its number or its alias, too.

You'll find the Quicksearch box in Bugzilla's footer area. On Bugzilla's front page, there is an additional Help[6] link which details how to use it.

Case Sensitivity in Searches

Bugzilla queries are case-insensitive and accent-insensitive, when used with either MySQL or Oracle databases. When using Bugzilla with PostgreSQL, however, some queries are case-sensitive. This is due to the way PostgreSQL handles case and accent sensitivity.

Bug Lists

If you run a search, a list of matching bugs will be returned.

The format of the list is configurable. For example, it can be sorted by clicking the column headings. Other useful features can be accessed using the links at the bottom of the list:

Long Format: this gives you a large page with a non-editable summary of the fields of each bug.

XML: get the buglist in the XML format.

CSV: get the buglist as comma-separated values, for import into e.g. a spreadsheet.

Feed: get the buglist as an Atom feed. Copy this link into your favorite feed reader. If you are using Firefox, you can also save the list as a live bookmark by clicking the live bookmark icon in the status bar. To limit the number of bugs in the feed, add a limit=n parameter to the URL.

iCalendar: Get the buglist as an iCalendar file. Each bug is represented as a to-do item in the imported calendar.

Change Columns: change the bug attributes which appear in the list.

Change several bugs at once: If your account is sufficiently empowered, and more than one bug appear in the bug list, this link is displayed which lets you make the same change to all the bugs in the list - for example, changing their assignee.

Send mail to bug assignees: If more than one bug appear in the bug list and there are at least two distinct bug assignees, this links is displayed which lets you easily send a mail to the assignees of all bugs on the list.

[6] `../../page.cgi?id=quicksearch.html`

Edit Search: If you didn't get exactly the results you were looking for, you can return to the Query page through this link and make small revisions to the query you just made so you get more accurate results.

Remember Search As: You can give a search a name and remember it; a link will appear in your page footer giving you quick access to run it again later.

Adding/removing tags to/from bugs

You can add and remove tags from individual bugs, which let you find and manage bugs more easily. Tags are per-user and so are only visible and editable by the user who created them. You can then run queries using tags as a criteria, either by using the Advanced Search form, or simply by typing "tag:my_tag_name" in the QuickSearch box at the top (or bottom) of the page. Tags can also be displayed in buglists.

This feature is useful when you want to keep track of several bugs, but for different reasons. Instead of adding yourself to the CC list of all these bugs and mixing all these reasons, you can now store these bugs in separate lists, e.g. "Keep in mind", "Interesting bugs", or "Triage". One big advantage of this way to manage bugs is that you can easily add or remove tags from bugs one by one.

Filing Bugs

Reporting a New Bug

Years of bug writing experience has been distilled for your reading pleasure into the Bug Writing Guidelines[7]. While some of the advice is Mozilla-specific, the basic principles of reporting Reproducible, Specific bugs, isolating the Product you are using, the Version of the Product, the Component which failed, the Hardware Platform, and Operating System you were using at the time of the failure go a long way toward ensuring accurate, responsible fixes for the bug that bit you.

The procedure for filing a bug is as follows:

1. Click the "New" link available in the footer of pages, or the "Enter a new bug report" link displayed on the home page of the Bugzilla installation.

   Note
   If you want to file a test bug to see how Bugzilla works, you can do it on one of our test installations on Landfill[a].

   [a] http://landfill.bugzilla.org/bugzilla-4.4-branch/

2. You first have to select the product in which you found a bug.
3. You now see a form where you can specify the component (part of the product which is affected by the bug you discovered; if you have no idea, just select "General" if such a component exists), the version of the program you were using, the Operating System and platform your program is running on and the severity of the bug (if the bug you found crashes the program, it's probably a major or a critical bug; if it's a typo somewhere, that's something pretty minor; if it's something you would like to see implemented, then that's an enhancement).
4. You now have to give a short but descriptive summary of the bug you found. "My program is crashing all the time" is a very poor summary and doesn't help developers at all. Try something more meaningful or your bug will probably be ignored due to a lack of precision. The next step is to give a very detailed list of steps to reproduce the problem you encountered. Try to limit these steps to a minimum set required to reproduce the problem. This will make the life of developers easier, and the probability that they consider your bug in a reasonable timeframe will be much higher.

   Note
   Try to make sure that everything in the summary is also in the first comment. Summaries are often updated and this will ensure your original information is easily accessible.

[7] http://landfill.bugzilla.org/bugzilla-4.4-branch/page.cgi?id=bug-writing.html

5. As you file the bug, you can also attach a document (testcase, patch, or screenshot of the problem).
6. Depending on the Bugzilla installation you are using and the product in which you are filing the bug, you can also request developers to consider your bug in different ways (such as requesting review for the patch you just attached, requesting your bug to block the next release of the product, and many other product specific requests).
7. Now is a good time to read your bug report again. Remove all misspellings, otherwise your bug may not be found by developers running queries for some specific words, and so your bug would not get any attention. Also make sure you didn't forget any important information developers should know in order to reproduce the problem, and make sure your description of the problem is explicit and clear enough. When you think your bug report is ready to go, the last step is to click the "Commit" button to add your report into the database.

You do not need to put "any" or similar strings in the URL field. If there is no specific URL associated with the bug, leave this field blank.

If you feel a bug you filed was incorrectly marked as a DUPLICATE of another, please question it in your bug, not the bug it was duped to. Feel free to CC the person who duped it if they are not already CCed.

Clone an Existing Bug

Starting with version 2.20, Bugzilla has a feature that allows you to clone an existing bug. The newly created bug will inherit most settings from the old bug. This allows you to track more easily similar concerns in a new bug. To use this, go to the bug that you want to clone, then click the "Clone This Bug" link on the bug page. This will take you to the "Enter Bug" page that is filled with the values that the old bug has. You can change those values and/or texts if needed.

Attachments

You should use attachments, rather than comments, for large chunks of ASCII data, such as trace, debugging output files, or log files. That way, it doesn't bloat the bug for everyone who wants to read it, and cause people to receive fat, useless mails.

You should make sure to trim screenshots. There's no need to show the whole screen if you are pointing out a single-pixel problem.

Bugzilla stores and uses a Content-Type for each attachment (e.g. text/html). To download an attachment as a different Content-Type (e.g. application/xhtml+xml), you can override this using a 'content_type' parameter on the URL, e.g. `&content_type=text/plain`.

Also, you can enter the URL pointing to the attachment instead of uploading the attachment itself. For example, this is useful if you want to point to an external application, a website or a very large file. Note that there is no guarantee that the source file will always be available, nor that its content will remain unchanged.

Another way to attach data is to paste text directly in the text field, and Bugzilla will convert it into an attachment. This is pretty useful when you do copy and paste, and you don't want to put the text in a temporary file first.

Patch Viewer

Viewing and reviewing patches in Bugzilla is often difficult due to lack of context, improper format and the inherent readability issues that raw patches present. Patch Viewer is an enhancement to Bugzilla designed to fix that by offering increased context, linking to sections, and integrating with Bonsai, LXR and CVS.

Patch viewer allows you to:

View patches in color, with side-by-side view rather than trying to interpret the contents of the patch.
See the difference between two patches.
Get more context in a patch.
Collapse and expand sections of a patch for easy reading.
Link to a particular section of a patch for discussion or review
Go to Bonsai or LXR to see more context, blame, and cross-references for the part of the patch you are looking at
Create a rawtext unified format diff out of any patch, no matter what format it came from

Viewing Patches in Patch Viewer

The main way to view a patch in patch viewer is to click on the "Diff" link next to a patch in the Attachments list on a bug. You may also do this within the edit window by clicking the "View Attachment As Diff" button in the Edit Attachment screen.

Seeing the Difference Between Two Patches

To see the difference between two patches, you must first view the newer patch in Patch Viewer. Then select the older patch from the dropdown at the top of the page ("Differences between [dropdown] and this patch") and click the "Diff" button. This will show you what is new or changed in the newer patch.

Getting More Context in a Patch

To get more context in a patch, you put a number in the textbox at the top of Patch Viewer ("Patch / File / [textbox]") and hit enter. This will give you that many lines of context before and after each change. Alternatively, you can click on the "File" link there and it will show each change in the full context of the file. This feature only works against files that were diffed using "cvs diff".

Collapsing and Expanding Sections of a Patch

To view only a certain set of files in a patch (for example, if a patch is absolutely huge and you want to only review part of it at a time), you can click the "(+)" and "(-)" links next to each file (to expand it or collapse it). If you want to collapse all files or expand all files, you can click the "Collapse All" and "Expand All" links at the top of the page.

Linking to a Section of a Patch

To link to a section of a patch (for example, if you want to be able to give someone a URL to show them which part you are talking about) you simply click the "Link Here" link on the section header. The resulting URL can be copied and used in discussion.

Going to Bonsai and LXR

To go to Bonsai to get blame for the lines you are interested in, you can click the "Lines XX-YY" link on the section header you are interested in. This works even if the patch is against an old version of the file, since Bonsai stores all versions of the file.

To go to LXR, you click on the filename on the file header (unfortunately, since LXR only does the most recent version, line numbers are likely to rot).

Creating a Unified Diff

If the patch is not in a format that you like, you can turn it into a unified diff format by clicking the "Raw Unified" link at the top of the page.

Hints and Tips

This section distills some Bugzilla tips and best practices that have been developed.

Autolinkification

Bugzilla comments are plain text - so typing <U> will produce less-than, U, greater-than rather than underlined text. However, Bugzilla will automatically make hyperlinks out of certain sorts of text in comments. For example, the text "http://www.bugzilla.org" will be turned into a link: http://www.bugzilla.org. Other strings which get linkified in the obvious manner are:

bug 12345
comment 7
bug 23456, comment 53
attachment 4321
mailto:george@example.com
george@example.com
ftp://ftp.mozilla.org
Most other sorts of URL

A corollary here is that if you type a bug number in a comment, you should put the word "bug" before it, so it gets autolinkified for the convenience of others.

Comments

If you are changing the fields on a bug, only comment if either you have something pertinent to say, or Bugzilla requires it. Otherwise, you may spam people unnecessarily with bug mail. To take an example: a user can set up their account to filter out messages where someone just adds themselves to the CC field of a bug (which happens a lot.) If you come along, add yourself to the CC field, and add a comment saying "Adding self to CC", then that person gets a pointless piece of mail they would otherwise have avoided.

Don't use sigs in comments. Signing your name ("Bill") is acceptable, if you do it out of habit, but full mail/news-style four line ASCII art creations are not.

Server-Side Comment Wrapping

Bugzilla stores comments unwrapped and wraps them at display time. This ensures proper wrapping in all browsers. Lines beginning with the ">" character are assumed to be quotes, and are not wrapped.

Dependency Tree

On the "Dependency tree" page linked from each bug page, you can see the dependency relationship from the bug as a tree structure.

You can change how much depth to show, and you can hide resolved bugs from this page. You can also collaps/expand dependencies for each bug on the tree view, using the [-]/[+] buttons that appear before its summary. This option is not available for terminal bugs in the tree (that don't have further dependencies).

Time Tracking Information

Users who belong to the group specified by the "timetrackinggroup" parameter have access to time-related fields. Developers can see deadlines and estimated times to fix bugs, and can provide time spent on these bugs.

At any time, a summary of the time spent by developers on bugs is accessible either from bug lists when clicking the "Time Summary" button or from individual bugs when clicking the "Summarize time" link in the time tracking table. The `summarize_time.cgi` page lets you view this information either per developer or per bug, and can be split on a month basis to have greater details on how time is spent by developers.

As soon as a bug is marked as RESOLVED, the remaining time expected to fix the bug is set to zero. This lets QA people set it again for their own usage, and it will be set to zero again when the bug will be marked as CLOSED.

User Preferences

Once logged in, you can customize various aspects of Bugzilla via the "Preferences" link in the page footer. The preferences are split into five tabs:

General Preferences

This tab allows you to change several default settings of Bugzilla.

- Bugzilla's general appearance (skin) - select which skin to use. Bugzilla supports adding custom skins.
- Quote the associated comment when you click on its reply link - sets the behavior of the comment "Reply" link. Options include quoting the full comment, just reference the comment number, or turn the link off.
- Language used in email - select which language email will be sent in, from the list of available languages.
- After changing a bug - This controls what page is displayed after changes to a bug are submitted. The options include to show the bug just modified, to show the next bug in your list, or to do nothing.
- Zoom textareas large when in use (requires JavaScript) - enable or disable the automatic expanding of text areas when text is being entered into them.
- Field separator character for CSV files - Select between a comma and semi-colon for exported CSV bug lists.
- Automatically add me to the CC list of bugs I change - set default behavior of CC list. Options include "Always", "Never", and "Only if I have no role on them".
- When viewing a bug, show comments in this order - controls the order of comments. Options include "Oldest to Newest", "Newest to Oldest" and "Newest to Oldest, but keep the bug description at the top".
- Show a quip at the top of each bug list - controls whether a quip will be shown on the Bug list page.

Email Preferences

This tab allows you to enable or disable email notification on specific events.

In general, users have almost complete control over how much (or how little) email Bugzilla sends them. If you want to receive the maximum amount of email possible, click the "Enable All Mail" button. If you don't want to receive any email from Bugzilla at all, click the "Disable All Mail" button.

Note
A Bugzilla administrator can stop a user from receiving bugmail by clicking the "Bugmail Disabled" checkbox when editing the user account. This is a drastic step best taken only for disabled accounts, as it overrides the user's individual mail preferences.

There are two global options -- "Email me when someone asks me to set a flag" and "Email me when someone sets a flag I asked for". These define how you want to receive bugmail with regards to flags. Their use is quite straightforward; enable the checkboxes if you want Bugzilla to send you mail under either of the above conditions.

If you'd like to set your bugmail to something besides 'Completely ON' and 'Completely OFF', the "Field/recipient specific options" table allows you to do just that. The rows of the table define events that can happen to a bug -- things like attachments being added, new comments being made, the priority changing, etc. The columns in the table define your relationship with the bug:

- Reporter - Where you are the person who initially reported the bug. Your name/account appears in the "Reporter:" field.
- Assignee - Where you are the person who has been designated as the one responsible for the bug. Your name/account appears in the "Assigned To:" field of the bug.
- QA Contact - You are one of the designated QA Contacts for the bug. Your account appears in the "QA Contact:" text-box of the bug.

- CC - You are on the list CC List for the bug. Your account appears in the “CC:” text box of the bug.
- Voter - You have placed one or more votes for the bug. Your account appears only if someone clicks on the “Show votes for this bug” link on the bug.

Note
Some columns may not be visible for your installation, depending on your site’s configuration.

To fine-tune your bugmail, decide the events for which you want to receive bugmail; then decide if you want to receive it all the time (enable the checkbox for every column), or only when you have a certain relationship with a bug (enable the checkbox only for those columns). For example: if you didn’t want to receive mail when someone added themselves to the CC list, you could uncheck all the boxes in the “CC Field Changes” line. As another example, if you never wanted to receive email on bugs you reported unless the bug was resolved, you would un-check all boxes in the “Reporter” column except for the one on the “The bug is resolved or verified” row.

Note
Bugzilla adds the “X-Bugzilla-Reason” header to all bugmail it sends, describing the recipient’s relationship (AssignedTo, Reporter, QAContact, CC, or Voter) to the bug. This header can be used to do further client-side filtering.

Bugzilla has a feature called “Users Watching”. When you enter one or more comma-delineated user accounts (usually email addresses) into the text entry box, you will receive a copy of all the bugmail those users are sent (security settings permitting). This powerful functionality enables seamless transitions as developers change projects or users go on holiday.

Note
The ability to watch other users may not be available in all Bugzilla installations. If you don’t see this feature, and feel that you need it, speak to your administrator.

Each user listed in the “Users watching you” field has you listed in their “Users to watch” list and can get bugmail according to your relationship to the bug and their “Field/recipient specific options” setting.

Saved Searches

On this tab you can view and run any Saved Searches that you have created, and also any Saved Searches that other members of the group defined in the "querysharegroup" parameter have shared. Saved Searches can be added to the page footer from this screen. If somebody is sharing a Search with a group she or he is allowed to assign users to, the sharer may opt to have the Search show up in the footer of the group’s direct members by default.

Name and Password

On this tab, you can change your basic account information, including your password, email address and real name. For security reasons, in order to change anything on this page you must type your *current* password into the “Password” field at the top of the page. If you attempt to change your email address, a confirmation email is sent to both the old and new addresses, with a link to use to confirm the change. This helps to prevent account hijacking.

Permissions

This is a purely informative page which outlines your current permissions on this installation of Bugzilla.

A complete list of permissions is below. Only users with *editusers* privileges can change the permissions of other users.

admin Indicates user is an Administrator.

bz_canusewhineatothers Indicates user can configure whine reports for other users.

bz_canusewhines Indicates user can configure whine reports for self.

bz_quip_moderators Indicates user can moderate quips.

bz_sudoers Indicates user can perform actions as other users.

bz_sudo_protect Indicates user cannot be impersonated by other users.

canconfirm Indicates user can confirm a bug or mark it a duplicate.

creategroups Indicates user can create and destroy groups.

editbugs Indicates user can edit all bug fields.

editclassifications Indicates user can create, destroy, and edit classifications.

editcomponents Indicates user can create, destroy, and edit components.

editkeywords Indicates user can create, destroy, and edit keywords.

editusers Indicates user can edit or disable users.

tweakparams Indicates user can change Parameters.

Note
For more information on how permissions work in Bugzilla (i.e. who can change what), see Section 6.4.

Reports and Charts

As well as the standard buglist, Bugzilla has two more ways of viewing sets of bugs. These are the reports (which give different views of the current state of the database) and charts (which plot the changes in particular sets of bugs over time.)

Reports

A report is a view of the current state of the bug database.

You can run either an HTML-table-based report, or a graphical line/pie/bar-chart-based one. The two have different pages to define them, but are close cousins - once you've defined and viewed a report, you can switch between any of the different views of the data at will.

Both report types are based on the idea of defining a set of bugs using the standard search interface, and then choosing some aspect of that set to plot on the horizontal and/or vertical axes. You can also get a form of 3-dimensional report by choosing to have multiple images or tables.

So, for example, you could use the search form to choose "all bugs in the WorldControl product", and then plot their severity against their component to see which component had had the largest number of bad bugs reported against it.

Once you've defined your parameters and hit "Generate Report", you can switch between HTML, CSV, Bar, Line and Pie. (Note: Pie is only available if you didn't define a vertical axis, as pie charts don't have one.) The other controls are fairly self-explanatory; you can change the size of the image if you find text is overwriting other text, or the bars are too thin to see.

Charts

A chart is a view of the state of the bug database over time.

Bugzilla currently has two charting systems - Old Charts and New Charts. Old Charts have been part of Bugzilla for a long time; they chart each status and resolution for each product, and that's all. They are deprecated, and going away soon - we won't say any more about them. New Charts are the future - they allow you to chart anything you can define as a search.

Note
Both charting forms require the administrator to set up the data-gathering script. If you can't see any charts, ask them whether they have done so.

An individual line on a chart is called a data set. All data sets are organised into categories and subcategories. The data sets that Bugzilla defines automatically use the Product name as a Category and Component names as Subcategories, but there is no need for you to follow that naming scheme with your own charts if you don't want to.

Data sets may be public or private. Everyone sees public data sets in the list, but only their creator sees private data sets. Only administrators can make data sets public. No two data sets, even two private ones, can have the same set of category, subcategory and name. So if you are creating private data sets, one idea is to have the Category be your username.

Creating Charts

You create a chart by selecting a number of data sets from the list, and pressing Add To List for each. In the List Of Data Sets To Plot, you can define the label that data set will have in the chart's legend, and also ask Bugzilla to Sum a number of data sets (e.g. you could Sum data sets representing RESOLVED, VERIFIED and CLOSED in a particular product to get a data set representing all the resolved bugs in that product.)

If you've erroneously added a data set to the list, select it using the checkbox and click Remove. Once you add more than one data set, a "Grand Total" line automatically appears at the bottom of the list. If you don't want this, simply remove it as you would remove any other line.

You may also choose to plot only over a certain date range, and to cumulate the results - that is, to plot each one using the previous one as a baseline, so the top line gives a sum of all the data sets. It's easier to try than to explain :-)

Once a data set is in the list, one can also perform certain actions on it. For example, one can edit the data set's parameters (name, frequency etc.) if it's one you created or if you are an administrator.

Once you are happy, click Chart This List to see the chart.

Creating New Data Sets

You may also create new data sets of your own. To do this, click the "create a new data set" link on the Create Chart page. This takes you to a search-like interface where you can define the search that Bugzilla will plot. At the bottom of the page, you choose the category, sub-category and name of your new data set.

If you have sufficient permissions, you can make the data set public, and reduce the frequency of data collection to less than the default seven days.

Flags

A flag is a kind of status that can be set on bugs or attachments to indicate that the bugs/attachments are in a certain state. Each installation can define its own set of flags that can be set on bugs or attachments.

If your installation has defined a flag, you can set or unset that flag, and if your administrator has enabled requesting of flags, you can submit a request for another user to set the flag.

To set a flag, select either "+" or "-" from the drop-down menu next to the name of the flag in the "Flags" list. The meaning of these values are flag-specific and thus cannot be described in this documentation, but by way of example, setting a flag named "review" to "+" may indicate that the bug/attachment has passed review, while setting it to "-" may indicate that the bug/attachment has failed review.

To unset a flag, click its drop-down menu and select the blank value. Note that marking an attachment as obsolete automatically cancels all pending requests for the attachment.

If your administrator has enabled requests for a flag, request a flag by selecting "?" from the drop-down menu and then entering the username of the user you want to set the flag in the text field next to the menu.

A set flag appears in bug reports and on "edit attachment" pages with the abbreviated username of the user who set the flag prepended to the flag name. For example, if Jack sets a "review" flag to "+", it appears as Jack: review [+]

A requested flag appears with the user who requested the flag prepended to the flag name and the user who has been requested to set the flag appended to the flag name within parentheses. For example, if Jack asks Jill for review, it appears as Jack: review [?] (Jill).

You can browse through open requests made of you and by you by selecting 'My Requests' from the footer. You can also look at open requests limited by other requesters, requestees, products, components, and flag names from this page. Note that you can use '-' for requestee to specify flags with 'no requestee' set.

Whining

Whining is a feature in Bugzilla that can regularly annoy users at specified times. Using this feature, users can execute saved searches at specific times (i.e. the 15th of the month at midnight) or at regular intervals (i.e. every 15 minutes on Sundays). The results of the searches are sent to the user, either as a single email or as one email per bug, along with some descriptive text.

Warning
Throughout this section it will be assumed that all users are members of the bz_canusewhines group, membership in which is required in order to use the Whining system. You can easily make all users members of the bz_canusewhines group by setting the User RegExp to ".*" (without the quotes).
Also worth noting is the bz_canusewhineatothers group. Members of this group can create whines for any user or group in Bugzilla using a extended form of the whining interface. Features only available to members of the bz_canusewhineatothers group will be noted in the appropriate places.

Note
For whining to work, a special Perl script must be executed at regular intervals. More information on this is available in Section 2.3.3.

Note
This section does not cover the whineatnews.pl script. See Section 2.3.2 for more information on The Whining Cron.

The Event

The whining system defines an "Event" as one or more queries being executed at regular intervals, with the results of said queries (if there are any) being emailed to the user. Events are created by clicking on the "Add new event" button.

Once a new event is created, the first thing to set is the "Email subject line". The contents of this field will be used in the subject line of every email generated by this event. In addition to setting a subject, space is provided to enter some descriptive text that will be included at the top of each message (to help you in understanding why you received the email in the first place).

The next step is to specify when the Event is to be run (the Schedule) and what searches are to be performed (the Searches).

Whining Schedule

Each whining event is associated with zero or more schedules. A schedule is used to specify when the query (specified below) is to be run. A new event starts out with no schedules (which means it will never run, as it is not scheduled to run). To add a schedule, press the "Add a new schedule" button.

Each schedule includes an interval, which you use to tell Bugzilla when the event should be run. An event can be run on certain days of the week, certain days of the month, during weekdays (defined as Monday through Friday), or every day.

Warning
Be careful if you set your event to run on the 29th, 30th, or 31st of the month, as your event may not run exactly when expected. If you want your event to run on the last day of the month, select "Last day of the month" as the interval.

Once you have specified the day(s) on which the event is to be run, you should now specify the time at which the event is to be run. You can have the event run at a certain hour on the specified day(s), or every hour, half-hour, or quarter-hour on the specified day(s).

If a single schedule does not execute an event as many times as you would want, you can create another schedule for the same event. For example, if you want to run an event on days whose numbers are divisible by seven, you would need to add four schedules to the event, setting the schedules to run on the 7th, 14th, 21st, and 28th (one day per schedule) at whatever time (or times) you choose.

Note
If you are a member of the bz_canusewhineatothers group, then you will be presented with another option: "Mail to". Using this you can control who will receive the emails generated by this event. You can choose to send the emails to a single user (identified by email address) or a single group (identified by group name). To send to multiple users or groups, create a new schedule for each additional user/group.

Whining Searches

Each whining event is associated with zero or more searches. A search is any saved search to be run as part of the specified schedule (see above). You start out without any searches associated with the event (which means that the event will not run, as there will never be any results to return). To add a search, press the "Include search" button.

The first field to examine in your newly added search is the Sort field. Searches are run, and results included, in the order specified by the Sort field. Searches with smaller Sort values will run before searches with bigger Sort values.

The next field to examine is the Search field. This is where you choose the actual search that is to be run. Instead of defining search parameters here, you are asked to choose from the list of saved searches (the same list that appears at the bottom of every Bugzilla page). You are only allowed to choose from searches that you have saved yourself (the default saved search, "My Bugs", is not a valid choice). If you do not have any saved searches, you can take this opportunity to create one (see Section 5.5.4).

Note
When running queries, the whining system acts as if you are the user executing the query. This means that the whining system will ignore bugs that match your query, but that you cannot access.

Once you have chosen the saved search to be executed, give the query a descriptive title. This title will appear in the email, above the results of the query. If you choose "One message per bug", the query title will appear at the top of each email that contains a bug matching your query.

Finally, decide if the results of the query should be sent in a single email, or if each bug should appear in its own email.

Warning
Think carefully before checking the "One message per bug" box. If you create a query that matches thousands of bugs, you will receive thousands of emails!

Saving Your Changes

Once you have defined at least one schedule, and created at least one query, go ahead and "Update/Commit". This will save your Event and make it available for immediate execution.

Note
If you ever feel like deleting your event, you may do so using the "Remove Event" button in the upper-right corner of each Event. You can also modify an existing event, so long as you "Update/Commit" after completing your modifications.

Chapter 6

Customizing Bugzilla

Bugzilla Extensions

One of the best ways to customize Bugzilla is by writing a Bugzilla Extension. Bugzilla Extensions let you modify both the code and UI of Bugzilla in a way that can be distributed to other Bugzilla users and ported forward to future versions of Bugzilla with minimal effort.

See the Bugzilla Extension documentation[1] for information on how to write an Extension.

Custom Skins

Bugzilla allows you to have multiple skins. These are custom CSS and possibly also custom images for Bugzilla. To create a new custom skin, you have two choices:

- Make a single CSS file, and put it in the `skins/contrib` directory.
- Make a directory that contains all the same CSS file names as `skins/standard/`, and put your directory in `skins/contrib/`.

After you put the file or the directory there, make sure to run checksetup.pl so that it can reset the file permissions correctly.

After you have installed the new skin, it will show up as an option in the user's General Preferences. If you would like to force a particular skin on all users, just select it in the Default Preferences and then uncheck "Enabled" on the preference.

Template Customization

Administrators can configure the look and feel of Bugzilla without having to edit Perl files or face the nightmare of massive merge conflicts when they upgrade to a newer version in the future.

Templatization also makes localized versions of Bugzilla possible, for the first time. It's possible to have Bugzilla's UI language determined by the user's browser. More information is available in Section 6.3.6.

Template Directory Structure

The template directory structure starts with top level directory named `template`, which contains a directory for each installed localization. The next level defines the language used in the templates. Bugzilla comes with English templates, so the directory name is `en`, and we will discuss `template/en` throughout the documentation. Below `template/en` is the `default` directory, which contains all the standard templates shipped with Bugzilla.

[1] `../html/api/Bugzilla/Extension.html`

Warning
A directory `data/templates` also exists; this is where Template Toolkit puts the compiled versions of the templates from either the default or custom directories. *Do not* directly edit the files in this directory, or all your changes will be lost the next time Template Toolkit recompiles the templates.

Choosing a Customization Method

If you want to edit Bugzilla's templates, the first decision you must make is how you want to go about doing so. There are two choices, and which you use depends mainly on the scope of your modifications, and the method you plan to use to upgrade Bugzilla.

The first method of making customizations is to directly edit the templates found in `template/en/default`. This is probably the best way to go about it if you are going to be upgrading Bugzilla through Bzr, because if you then execute a **bzr update**, any changes you have made will be merged automagically with the updated versions.

Note
If you use this method, and Bzr conflicts occur during an update, the conflicted templates (and possibly other parts of your installation) will not work until they are resolved.

The second method is to copy the templates to be modified into a mirrored directory structure under `template/en/custom`. Templates in this directory structure automatically override any identically-named and identically-located templates in the `default` directory.

Note
The `custom` directory does not exist at first and must be created if you want to use it.

The second method of customization should be used if you use the overwriting method of upgrade, because otherwise your changes will be lost. This method may also be better if you are using the Bzr method of upgrading and are going to make major changes, because it is guaranteed that the contents of this directory will not be touched during an upgrade, and you can then decide whether to continue using your own templates, or make the effort to merge your changes into the new versions by hand.

Using this method, your installation may break if incompatible changes are made to the template interface. Such changes should be documented in the release notes, provided you are using a stable release of Bugzilla. If you use using unstable code, you will need to deal with this one yourself, although if possible the changes will be mentioned before they occur in the deprecations section of the previous stable release's release notes.

Note
Regardless of which method you choose, it is recommended that you run **./checksetup.pl** after editing any templates in the `template/en/default` directory, and after creating or editing any templates in the `custom` directory.

Warning
It is *required* that you run **./checksetup.pl** after creating a new template in the `custom` directory. Failure to do so will raise an incomprehensible error message.

How To Edit Templates

Note
If you are making template changes that you intend on submitting back for inclusion in standard Bugzilla, you should read the relevant sections of the Developers' Guide[a].

[a] http://www.bugzilla.org/docs/developer.html

The syntax of the Template Toolkit language is beyond the scope of this guide. It's reasonably easy to pick up by looking at the current templates; or, you can read the manual, available on the Template Toolkit home page[2].

One thing you should take particular care about is the need to properly HTML filter data that has been passed into the template. This means that if the data can possibly contain special HTML characters such as <, and the data was not intended to be HTML, they need to be converted to entity form, i.e. <. You use the 'html' filter in the Template Toolkit to do this (or the 'uri' filter to encode special characters in URLs). If you forget, you may open up your installation to cross-site scripting attacks.

Editing templates is a good way of doing a "poor man's custom fields". For example, if you don't use the Status Whiteboard, but want to have a free-form text entry box for "Build Identifier", then you can just edit the templates to change the field labels. It's still be called status_whiteboard internally, but your users don't need to know that.

Template Formats and Types

Some CGI's have the ability to use more than one template. For example, `buglist.cgi` can output itself as RDF, or as two formats of HTML (complex and simple). The mechanism that provides this feature is extensible.

Bugzilla can support different types of output, which again can have multiple formats. In order to request a certain type, you can append the &ctype=<contenttype> (such as rdf or html) to the `<cginame>.cgi` URL. If you would like to retrieve a certain format, you can use the &format=<format> (such as simple or complex) in the URL.

To see if a CGI supports multiple output formats and types, grep the CGI for "get_format". If it's not present, adding multiple format/type support isn't too hard - see how it's done in other CGIs, e.g. config.cgi.

To make a new format template for a CGI which supports this, open a current template for that CGI and take note of the INTERFACE comment (if present.) This comment defines what variables are passed into this template. If there isn't one, I'm afraid you'll have to read the template and the code to find out what information you get.

Write your template in whatever markup or text style is appropriate.

You now need to decide what content type you want your template served as. The content types are defined in the `Bugzilla/Constants.pm` file in the `contenttypes` constant. If your content type is not there, add it. Remember the three- or four-letter tag assigned to your content type. This tag will be part of the template filename.

Note
After adding or changing a content type, it's suitable to edit `Bugzilla/Constants.pm` in order to reflect the changes. Also, the file should be kept up to date after an upgrade if content types have been customized in the past.

Save the template as `<stubname>-<formatname>.<contenttypetag>.tmpl`. Try out the template by calling the CGI as `<cginame>.cgi?format=<formatname>&ctype=<type>`.

Particular Templates

There are a few templates you may be particularly interested in customizing for your installation.

index.html.tmpl: This is the Bugzilla front page.

[2] http://www.template-toolkit.org

global/header.html.tmpl: This defines the header that goes on all Bugzilla pages. The header includes the banner, which is what appears to users and is probably what you want to edit instead. However the header also includes the HTML HEAD section, so you could for example add a stylesheet or META tag by editing the header.

global/banner.html.tmpl: This contains the "banner", the part of the header that appears at the top of all Bugzilla pages. The default banner is reasonably barren, so you'll probably want to customize this to give your installation a distinctive look and feel. It is recommended you preserve the Bugzilla version number in some form so the version you are running can be determined, and users know what docs to read.

global/footer.html.tmpl: This defines the footer that goes on all Bugzilla pages. Editing this is another way to quickly get a distinctive look and feel for your Bugzilla installation.

global/variables.none.tmpl: This defines a list of terms that may be changed in order to "brand" the Bugzilla instance In this way, terms like "bugs" can be replaced with "issues" across the whole Bugzilla installation. The name "Bugzilla" and other words can be customized as well.

list/table.html.tmpl: This template controls the appearance of the bug lists created by Bugzilla. Editing this template allows per-column control of the width and title of a column, the maximum display length of each entry, and the wrap behaviour of long entries. For long bug lists, Bugzilla inserts a 'break' every 100 bugs by default; this behaviour is also controlled by this template, and that value can be modified here.

bug/create/user-message.html.tmpl: This is a message that appears near the top of the bug reporting page. By modifying this, you can tell your users how they should report bugs.

bug/process/midair.html.tmpl: This is the page used if two people submit simultaneous changes to the same bug. The second person to submit their changes will get this page to tell them what the first person did, and ask if they wish to overwrite those changes or go back and revisit the bug. The default title and header on this page read "Mid-air collision detected!" If you work in the aviation industry, or other environment where this might be found offensive (yes, we have true stories of this happening) you'll want to change this to something more appropriate for your environment.

bug/create/create.html.tmpl and **bug/create/comment.txt.tmpl**: You may not wish to go to the effort of creating custom fields in Bugzilla, yet you want to make sure that each bug report contains a number of pieces of important information for which there is not a special field. The bug entry system has been designed in an extensible fashion to enable you to add arbitrary HTML widgets, such as drop-down lists or textboxes, to the bug entry page and have their values appear formatted in the initial comment. A hidden field that indicates the format should be added inside the form in order to make the template functional. Its value should be the suffix of the template filename. For example, if the file is called `create-cust.html.tmpl`, then

```
<input type="hidden" name="format" value="cust">
```

should be used inside the form.

An example of this is the mozilla.org guided bug submission form[3]. The code for this comes with the Bugzilla distribution as an example for you to copy. It can be found in the files `create-guided.html.tmpl` and `comment-guided.html.tmpl`.

So to use this feature, create a custom template for `enter_bug.cgi`. The default template, on which you could base it, is `custom/bug/create/create.html.tmpl`. Call it `create-<formatname>.html.tmpl`, and in it, add widgets for each piece of information you'd like collected - such as a build number, or set of steps to reproduce.

Then, create a template like `custom/bug/create/comment.txt.tmpl`, and call it `comment-<formatname>.txt.tmpl`. This template should reference the form fields you have created using the syntax `[% form.<fieldname> %]`. When a bug report is submitted, the initial comment attached to the bug report will be formatted according to the layout of this template.

For example, if your custom enter_bug template had a field

```
<input type="text" name="buildid" size="30">
```

and then your comment.txt.tmpl had

```
BuildID: [% form.buildid %]
```

then something like

```
BuildID: 20020303
```

would appear in the initial comment.

[3] http://landfill.bugzilla.org/bugzilla-tip/enter_bug.cgi?product=WorldControl;format=guided

Configuring Bugzilla to Detect the User's Language

Bugzilla honours the user's Accept: HTTP header. You can install templates in other languages, and Bugzilla will pick the most appropriate according to a priority order defined by you. Many language templates can be obtained from http://www.bugzilla.org/download.html#localizations. Instructions for submitting new languages are also available from that location.

Customizing Who Can Change What

Warning
This feature should be considered experimental; the Bugzilla code you will be changing is not stable, and could change or move between versions. Be aware that if you make modifications as outlined here, you may have to re-make them or port them if Bugzilla changes internally between versions, and you upgrade.

Companies often have rules about which employees, or classes of employees, are allowed to change certain things in the bug system. For example, only the bug's designated QA Contact may be allowed to VERIFY the bug. Bugzilla has been designed to make it easy for you to write your own custom rules to define who is allowed to make what sorts of value transition.

By default, assignees, QA owners and users with *editbugs* privileges can edit all fields of bugs, except group restrictions (unless they are members of the groups they are trying to change). Bug reporters also have the ability to edit some fields, but in a more restrictive manner. Other users, without *editbugs* privileges, cannot edit bugs, except to comment and add themselves to the CC list.

For maximum flexibility, customizing this means editing Bugzilla's Perl code. This gives the administrator complete control over exactly who is allowed to do what. The relevant method is called `check_can_change_field()`, and is found in `Bug.pm` in your Bugzilla/ directory. If you open that file and search for "sub check_can_change_field", you'll find it.

This function has been carefully commented to allow you to see exactly how it works, and give you an idea of how to make changes to it. Certain marked sections should not be changed - these are the "plumbing" which makes the rest of the function work. In between those sections, you'll find snippets of code like:

```
# Allow the assignee to change anything.
if ($ownerid eq $whoid) {
    return 1;
}
```

It's fairly obvious what this piece of code does.

So, how does one go about changing this function? Well, simple changes can be made just by removing pieces - for example, if you wanted to prevent any user adding a comment to a bug, just remove the lines marked "Allow anyone to change comments." If you don't want the Reporter to have any special rights on bugs they have filed, just remove the entire section that deals with the Reporter.

More complex customizations are not much harder. Basically, you add a check in the right place in the function, i.e. after all the variables you are using have been set up. So, don't look at $ownerid before $ownerid has been obtained from the database. You can either add a positive check, which returns 1 (allow) if certain conditions are true, or a negative check, which returns 0 (deny.) E.g.:

```
if ($field eq "qacontact") {
    if (Bugzilla->user->in_group("quality_assurance")) {
        return 1;
    }
    else {
        return 0;
    }
}
```

This says that only users in the group "quality_assurance" can change the QA Contact field of a bug.

Getting more weird:

```
if (($field eq "priority") &&
    (Bugzilla->user->email =~ /.*\@example\.com$/))
{
    if ($oldvalue eq "P1") {
        return 1;
    }
    else {
        return 0;
    }
}
```

This says that if the user is trying to change the priority field, and their email address is @example.com, they can only do so if the old value of the field was "P1". Not very useful, but illustrative.

Warning
If you are modifying `process_bug.cgi` in any way, do not change the code that is bounded by DO_NOT_CHANGE blocks. Doing so could compromise security, or cause your installation to stop working entirely.

For a list of possible field names, look at the bugs table in the database. If you need help writing custom rules for your organization, ask in the newsgroup.

Integrating Bugzilla with Third-Party Tools

Many utilities and applications can integrate with Bugzilla, either on the client- or server-side. None of them are maintained by the Bugzilla community, nor are they tested during our QA tests, so use them at your own risk. They are listed at https://wiki.mozilla.org/Bugzilla:Addons.

Chapter 7

Glossary

0-9, high ascii

.htaccess

Apache web server, and other NCSA-compliant web servers, observe the convention of using files in directories called `.htaccess` to restrict access to certain files. In Bugzilla, they are used to keep secret files which would otherwise compromise your installation - e.g. the `localconfig` file contains the password to your database. curious.

A

Apache

In this context, Apache is the web server most commonly used for serving up Bugzilla pages. Contrary to popular belief, the apache web server has nothing to do with the ancient and noble Native American tribe, but instead derived its name from the fact that it was "a patchy" version of the original NCSA world-wide-web server.

USEFUL DIRECTIVES WHEN CONFIGURING BUGZILLA

`AddHandler`[1] Tell Apache that it's OK to run CGI scripts.

`AllowOverride`[2], **`Options`**[3] These directives are used to tell Apache many things about the directory they apply to. For Bugzilla's purposes, we need them to allow script execution and `.htaccess` overrides.

`DirectoryIndex`[4] Used to tell Apache what files are indexes. If you can not add `index.cgi` to the list of valid files, you'll need to set `$index_html` to 1 in `localconfig` so **./checksetup.pl** will create an `index.html` that redirects to `index.cgi`.

`ScriptInterpreterSource`[5] Used when running Apache on windows so the shebang line doesn't have to be changed in every Bugzilla script.

For more information about how to configure Apache for Bugzilla, see Section 2.2.4.1.

B

Bug

A "bug" in Bugzilla refers to an issue entered into the database which has an associated number, assignments, comments, etc. Some also refer to a "tickets" or "issues"; in the context of Bugzilla, they are synonymous.

Bug Number

Each Bugzilla bug is assigned a number that uniquely identifies that bug. The bug associated with a bug number can be pulled up via a query, or easily from the very front page by typing the number in the "Find" box.

Bugzilla

Bugzilla is the world-leading free software bug tracking system.

C

Common Gateway Interface (`CGI`)

CGI is an acronym for Common Gateway Interface. This is a standard for interfacing an external application with a web server. Bugzilla is an example of a CGI application.

Component

A Component is a subsection of a Product. It should be a narrow category, tailored to your organization. All Products must contain at least one Component (and, as a matter of fact, creating a Product with no Components will create an error in Bugzilla).

Comprehensive Perl Archive Network (`CPAN`)

CPAN stands for the “Comprehensive Perl Archive Network”. CPAN maintains a large number of extremely useful *Perl* modules - encapsulated chunks of code for performing a particular task.

`contrib`

The `contrib` directory is a location to put scripts that have been contributed to Bugzilla but are not a part of the official distribution. These scripts are written by third parties and may be in languages other than perl. For those that are in perl, there may be additional modules or other requirements than those of the official distribution.

Note
Scripts in the `contrib` directory are not officially supported by the Bugzilla team and may break in between versions.

D

daemon

A daemon is a computer program which runs in the background. In general, most daemons are started at boot time via System V init scripts, or through RC scripts on BSD-based systems. *mysqld*, the MySQL server, and *apache*, a web server, are generally run as daemons.

DOS Attack

A DOS, or Denial of Service attack, is when a user attempts to deny access to a web server by repeatedly accessing a page or sending malformed requests to a webserver. A D-DOS, or Distributed Denial of Service attack, is when these requests come from multiple sources at the same time. Unfortunately, these are much more difficult to defend against.

G

Groups

The word “Groups” has a very special meaning to Bugzilla. Bugzilla’s main security mechanism comes by placing users in groups, and assigning those groups certain privileges to view bugs in particular *Products* in the *Bugzilla* database.

J

JavaScript

JavaScript is cool, we should talk about it.

M

Message Transport Agent (MTA)

A Message Transport Agent is used to control the flow of email on a system. The Email::Send[6] Perl module, which Bugzilla uses to send email, can be configured to use many different underlying implementations for actually sending the mail using the `mail_delivery_method` parameter.

MySQL

MySQL is one of the supported *RDBMS* for Bugzilla. MySQL can be downloaded from `http://www.mysql.com`. While you should familiarize yourself with all of the documentation, some high points are:

Backup[7] Methods for backing up your Bugzilla database.

Option Files[8] Information about how to configure MySQL using `my.cnf`.

Privilege System[9] Information about how to protect your MySQL server.

P

Perl Package Manager (PPM)

`http://aspn.activestate.com/ASPN/Downloads/ActivePerl/PPM/`

Product

A Product is a broad category of types of bugs, normally representing a single piece of software or entity. In general, there are several Components to a Product. A Product may define a group (used for security) for all bugs entered into its Components.

Perl

First written by Larry Wall, Perl is a remarkable program language. It has the benefits of the flexibility of an interpreted scripting language (such as shell script), combined with the speed and power of a compiled language, such as C. *Bugzilla* is maintained in Perl.

Q

QA

"QA", "Q/A", and "Q.A." are short for "Quality Assurance". In most large software development organizations, there is a team devoted to ensuring the product meets minimum standards before shipping. This team will also generally want to track the progress of bugs over their life cycle, thus the need for the "QA Contact" field in a bug.

[6] `http://search.cpan.org/dist/Email-Send/lib/Email/Send.pm`

R

Relational DataBase Management System (RDBMS)

A relational database management system is a database system that stores information in tables that are related to each other.

Regular Expression (regexp)

A regular expression is an expression used for pattern matching. Documentation[10]

S

Service

In Windows NT environment, a boot-time background application is referred to as a service. These are generally managed through the control panel while logged in as an account with "Administrator" level capabilities. For more information, consult your Windows manual or the MSKB.

SGML

SGML stands for "Standard Generalized Markup Language". Created in the 1980's to provide an extensible means to maintain documentation based upon content instead of presentation, SGML has withstood the test of time as a robust, powerful language. *XML* is the "baby brother" of SGML; any valid XML document it, by definition, a valid SGML document. The document you are reading is written and maintained in SGML, and is also valid XML if you modify the Document Type Definition.

T

Target Milestone

Target Milestones are Product goals. They are configurable on a per-Product basis. Most software development houses have a concept of "milestones" where the people funding a project expect certain functionality on certain dates. Bugzilla facilitates meeting these milestones by giving you the ability to declare by which milestone a bug will be fixed, or an enhancement will be implemented.

Tool Command Language (TCL)

TCL is an open source scripting language available for Windows, Macintosh, and Unix based systems. Bugzilla 1.0 was written in TCL but never released. The first release of Bugzilla was 2.0, which was when it was ported to perl.

Z

Zarro Boogs Found

This is just a goofy way of saying that there were no bugs found matching your query. When asked to explain this message, Terry had the following to say:

> I've been asked to explain this ... way back when, when Netscape released version 4.0 of its browser, we had a release party. Naturally, there had been a big push to try and fix every known bug before the release. Naturally, that hadn't actually happened. (This is not unique to Netscape or to 4.0; the same thing has happened with every software project I've ever seen.) Anyway, at the release party, T-shirts were handed out that said something like "Netscape 4.0: Zarro Boogs". Just like the software, the T-shirt had no known bugs. Uh-huh.
>
> So, when you query for a list of bugs, and it gets no results, you can think of this as a friendly reminder. Of *course* there are bugs matching your query, they just aren't in the bugsystem yet...
>
> —Terry Weissman

[10] http://perldoc.com/perl5.6/pod/perlre.html#Regular-Expressions

Appendix A

Troubleshooting

This section gives solutions to common Bugzilla installation problems. If none of the section headings seems to match your problem, read the general advice.

General Advice

If you can't get `checksetup.pl` to run to completion, it normally explains what's wrong and how to fix it. If you can't work it out, or if it's being uncommunicative, post the errors in the mozilla.support.bugzilla[1] newsgroup.

If you have made it all the way through Section 2.1 (Installation) and Section 2.2 (Configuration) but accessing the Bugzilla URL doesn't work, the first thing to do is to check your web server error log. For Apache, this is often located at `/etc/logs/httpd/error_log`. The error messages you see may be self-explanatory enough to enable you to diagnose and fix the problem. If not, see below for some commonly-encountered errors. If that doesn't help, post the errors to the newsgroup.

Bugzilla can also log all user-based errors (and many code-based errors) that occur, without polluting the web server's error log. To enable Bugzilla error logging, create a file that Bugzilla can write to, named `errorlog`, in the Bugzilla `data` directory. Errors will be logged as they occur, and will include the type of the error, the IP address and username (if available) of the user who triggered the error, and the values of all environment variables; if a form was being submitted, the data in the form will also be included. To disable error logging, delete or rename the `errorlog` file.

The Apache web server is not serving Bugzilla pages

After you have run **checksetup.pl** twice, run **testserver.pl http://yoursite.yourdomain/yoururl** to confirm that your web server is configured properly for Bugzilla.

```
bash$ ./testserver.pl http://landfill.bugzilla.org/bugzilla-tip
TEST-OK Webserver is running under group id in $webservergroup.
TEST-OK Got ant picture.
TEST-OK Webserver is executing CGIs.
TEST-OK Webserver is preventing fetch of http://landfill.bugzilla.org/bugzilla-tip/ ↩
    localconfig.
```

I installed a Perl module, but `checksetup.pl` claims it's not installed!

This could be caused by one of two things:

[1] news://news.mozilla.org/mozilla.support.bugzilla

1. You have two versions of Perl on your machine. You are installing modules into one, and Bugzilla is using the other. Rerun the CPAN commands (or manual compile) using the full path to Perl from the top of `checksetup.pl`. This will make sure you are installing the modules in the right place.
2. The permissions on your library directories are set incorrectly. They must, at the very least, be readable by the web server user or group. It is recommended that they be world readable.

DBD::Sponge::db prepare failed

The following error message may appear due to a bug in DBD::mysql (over which the Bugzilla team have no control):

```
DBD::Sponge::db prepare failed: Cannot determine NUM_OF_FIELDS at D:/Perl/site/lib/DBD/ ↩
    mysql.pm line 248.
 SV = NULL(0x0) at 0x20fc444
 REFCNT = 1
 FLAGS = (PADBUSY,PADMY)
```

To fix this, go to `<path-to-perl>/lib/DBD/sponge.pm` in your Perl installation and replace

```
my $numFields;
if ($attribs->{'NUM_OF_FIELDS'}) {
    $numFields = $attribs->{'NUM_OF_FIELDS'};
} elsif ($attribs->{'NAME'}) {
    $numFields = @{$attribs->{NAME}};
```

with

```
my $numFields;
if ($attribs->{'NUM_OF_FIELDS'}) {
    $numFields = $attribs->{'NUM_OF_FIELDS'};
} elsif ($attribs->{'NAMES'}) {
    $numFields = @{$attribs->{NAMES}};
```

(note the S added to NAME.)

cannot chdir(/var/spool/mqueue)

If you are installing Bugzilla on SuSE Linux, or some other distributions with "paranoid" security options, it is possible that the checksetup.pl script may fail with the error:

```
cannot chdir(/var/spool/mqueue): Permission denied
```

This is because your `/var/spool/mqueue` directory has a mode of `drwx------`. Type **chmod 755 /var/spool/mqueue** as root to fix this problem. This will allow any process running on your machine the ability to *read* the `/var/spool/mqueue` directory.

Everybody is constantly being forced to relogin

The most-likely cause is that the "cookiepath" parameter is not set correctly in the Bugzilla configuration. You can change this (if you're a Bugzilla administrator) from the editparams.cgi page via the web interface.

The value of the cookiepath parameter should be the actual directory containing your Bugzilla installation, *as seen by the end-user's web browser*. Leading and trailing slashes are mandatory. You can also set the cookiepath to any directory which is a parent of the Bugzilla directory (such as '/', the root directory). But you can't put something that isn't at least a partial match or it won't work. What you're actually doing is restricting the end-user's browser to sending the cookies back only to that directory.

How do you know if you want your specific Bugzilla directory or the whole site?

If you have only one Bugzilla running on the server, and you don't mind having other applications on the same server with it being able to see the cookies (you might be doing this on purpose if you have other things on your site that share authentication with Bugzilla), then you'll want to have the cookiepath set to "/", or to a sufficiently-high enough directory that all of the involved apps can see the cookies.

Example A.1 Examples of urlbase/cookiepath pairs for sharing login cookies

```
urlbase is http://bugzilla.mozilla.org/
cookiepath is /

urlbase is http://tools.mysite.tld/bugzilla/
        but you have http://tools.mysite.tld/someotherapp/ which shares
        authentication with your Bugzilla
cookiepath is /
```

On the other hand, if you have more than one Bugzilla running on the server (some people do - we do on landfill) then you need to have the cookiepath restricted enough so that the different Bugzillas don't confuse their cookies with one another.

Example A.2 Examples of urlbase/cookiepath pairs to restrict the login cookie

```
urlbase is http://landfill.bugzilla.org/bugzilla-tip/
cookiepath is /bugzilla-tip/

urlbase is http://landfill.bugzilla.org/bugzilla-4.0-branch/
cookiepath is /bugzilla-4.0-branch/
```

If you had cookiepath set to "/" at any point in the past and need to set it to something more restrictive (i.e. "/bugzilla/"), you can safely do this without requiring users to delete their Bugzilla-related cookies in their browser (this is true starting with Bugzilla 2.18 and Bugzilla 2.16.5).

`index.cgi` doesn't show up unless specified in the URL

You probably need to set up your web server in such a way that it will serve the index.cgi page as an index page.

If you are using Apache, you can do this by adding `index.cgi` to the end of the `DirectoryIndex` line as mentioned in Section 2.2.4.1.

checksetup.pl reports "Client does not support authentication protocol requested by server..."

This error is occurring because you are using the new password encryption that comes with MySQL 4.1, while your `DBD::mysql` module was compiled against an older version of MySQL. If you recompile `DBD::mysql` against the current MySQL libraries (or just obtain a newer version of this module) then the error may go away.

If that does not fix the problem, or if you cannot recompile the existing module (e.g. you're running Windows) and/or don't want to replace it (e.g. you want to keep using a packaged version), then a workaround is available from the MySQL docs: http://dev.mysql.com/doc/mysql/en/Old_client.html

Appendix B

Contrib

There are a number of unofficial Bugzilla add-ons in the `$BUGZILLA_ROOT/contrib/` directory. This section documents them.

Command-line Search Interface

There are a suite of Unix utilities for searching Bugzilla from the command line. They live in the `contrib/cmdline` directory. There are three files - `query.conf`, `buglist` and `bugs`.

Warning
These files pre-date the templatization work done as part of the 2.16 release, and have not been updated.

`query.conf` contains the mapping from options to field names and comparison types. Quoted option names are "grepped" for, so it should be easy to edit this file. Comments (#) have no effect; you must make sure these lines do not contain any quoted "option".

`buglist` is a shell script that submits a Bugzilla query and writes the resulting HTML page to stdout. It supports both short options, (such as "-Afoo" or "-Rbar") and long options (such as "--assignedto=foo" or "--reporter=bar"). If the first character of an option is not "-", it is treated as if it were prefixed with "--default=".

The column list is taken from the COLUMNLIST environment variable. This is equivalent to the "Change Columns" option that is available when you list bugs in buglist.cgi. If you have already used Bugzilla, grep for COLUMNLIST in your cookies file to see your current COLUMNLIST setting.

`bugs` is a simple shell script which calls `buglist` and extracts the bug numbers from the output. Adding the prefix "http://bugzilla.mozi turns the bug list into a working link if any bugs are found. Counting bugs is easy. Pipe the results through **sed -e 's/,/ /g' | wc | awk '{printf $2 "\n"}'**

Akkana Peck says she has good results piping `buglist` output through **w3m -T text/html -dump**

Command-line 'Send Unsent Bug-mail' tool

Within the `contrib` directory exists a utility with the descriptive (if compact) name of `sendunsentbugmail.pl`. The purpose of this script is, simply, to send out any bug-related mail that should have been sent by now, but for one reason or another has not.

To accomplish this task, `sendunsentbugmail.pl` uses the same mechanism as the `sanitycheck.cgi` script; it scans through the entire database looking for bugs with changes that were made more than 30 minutes ago, but where there is no record

of anyone related to that bug having been sent mail. Having compiled a list, it then uses the standard rules to determine who gets mail, and sends it out.

As the script runs, it indicates the bug for which it is currently sending mail; when it has finished, it gives a numerical count of how many mails were sent and how many people were excluded. (Individual user names are not recorded or displayed.) If the script produces no output, that means no unsent mail was detected.

Usage: move the sendunsentbugmail.pl script up into the main directory, ensure it has execute permission, and run it from the command line (or from a cron job) with no parameters.

Appendix C

Manual Installation of Perl Modules

Instructions

If you need to install Perl modules manually, here's how it's done. Download the module using the link given in the next section, and then apply this magic incantation, as root:

```
bash# tar -xzvf <module>.tar.gz
bash# cd <module>
bash# perl Makefile.PL
bash# make
bash# make test
bash# make install
```

Note
In order to compile source code under Windows you will need to obtain a 'make' utility. The **nmake** utility provided with Microsoft Visual C++ may be used. As an alternative, there is a utility called **dmake** available from CPAN which is written entirely in Perl. As described in Section C.2, however, most packages already exist and are available from ActiveState or theory58S. We highly recommend that you install them using the ppm GUI available with ActiveState and to add the theory58S repository to your list of repositories.

Download Locations

Note
Running Bugzilla on Windows requires the use of ActiveState Perl 5.8.1 or higher. Many modules already exist in the core distribution of ActiveState Perl. If some modules are missing, upgrade ActiveState Perl to at least 5.12; it has all the required modules.

CGI:

```
CPAN Download Page:
Documentation:
```

Data-Dumper:

```
CPAN Download Page:
Documentation:
```

Date::Format (part of TimeDate):

```
CPAN Download Page:
Documentation:
```

DBI:

```
CPAN Download Page:
Documentation:
```

DBD::mysql:

```
CPAN Download Page:
Documentation:
```

DBD::Pg:

```
CPAN Download Page:
Documentation:
```

Template-Toolkit:

```
CPAN Download Page:
Documentation:
```

GD:

```
CPAN Download Page:
Documentation:
```

Template::Plugin::GD:

```
CPAN Download Page:
Documentation:
```

MIME::Parser (part of MIME-tools):

```
CPAN Download Page:
Documentation:
```

Optional Modules

Chart::Lines:

```
CPAN Download Page:
Documentation:
```

GD::Graph:

```
CPAN Download Page:
Documentation:
```

GD::Text::Align (part of GD::Text::Util):

```
CPAN Download Page:
Documentation:
```

XML::Twig:

```
CPAN Download Page:
Documentation:
```

PatchReader:

```
CPAN Download Page:
Documentation:
```

Appendix D

GNU Free Documentation License

Version 1.1, March 2000

Preamble

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

Applicability and Definition

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

Verbatim Copying

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

Copying in Quantity

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

Modifications

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

Combining Documents

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

Collections of Documents

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

Aggregation with Independent Works

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

Translation

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

Termination

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

Future Revisions of this License

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.